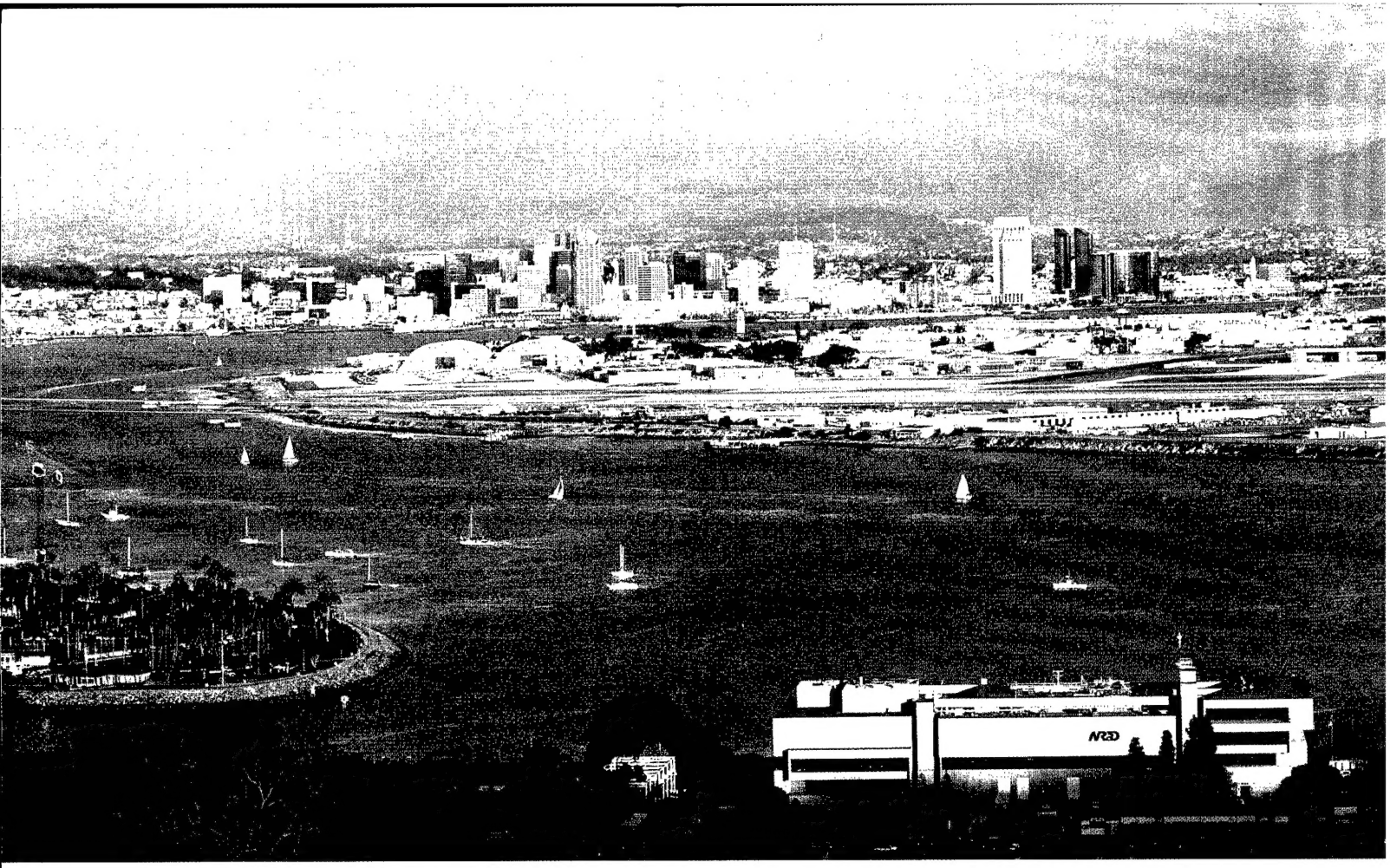


Technical Report 1722
June 1996

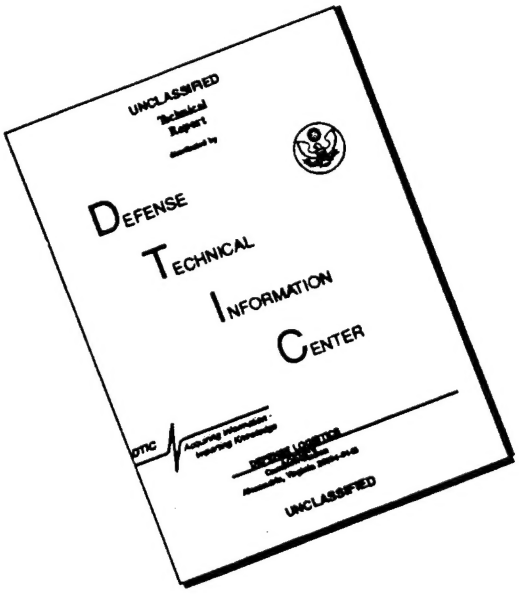
Synthetic Theater of War (STOW) Engineering Demonstration-1A (ED-1A) Analysis Report

T. R. Tiernan

Naval Command, Control and
Ocean Surveillance Center
RDT&E Division

San Diego, CA
92152-5001

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
RDT&E DIVISION
San Diego, California 92152-5001

H. A. WILLIAMS, CAPT, USN
Commanding Officer

R. C. KOLB
Acting Executive Director

ADMINISTRATIVE INFORMATION

The work detailed in this report was performed for the Defense Advanced Research Projects Agency by the Naval Command, Control and Ocean Surveillance Center RDT&E Division, Synthetic Battlespace Branch, Code 44203, and Simulation Infrastructure Branch, Code 44204. The branches received support from the Naval Research Laboratory, Advanced Telecommunications, Inc., Bolt, Beranek & Newman, Loral Advanced Distributed Simulation, and the Massachusetts Institute of Technology Lincoln Lab. Funding was provided under program element 0603226E.

Released by
C. B. Peters, Head
Synthetic Battlespace Branch

K. E. Boner, Head
Simulation Infrastructure Branch

Under authority of
J. D. Grossman, Head
Simulation and Human Systems
Technology Division

ADS PROGRAM MANAGER

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (ARPA)
Dr. Stuart Milner

PROGRAM MANAGER

NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
SIMULATION INFRASTRUCTURE
Thomas R. Tiernan

SYSTEMS ENGINEERING TEAM

NAVAL COMMAND, CONTROL AND OCEAN SURVEILLANCE CENTER
SIMULATION INTEGRATION PROJECT TEAM
Kevin E. Boner
Adam Whitlock
Cindy M. Keune
James Carlson

EXECUTIVE SUMMARY

The primary goal of the Real-Time Information Transfer and Networking (RITN) effort is to evaluate and develop technologies to integrate into a scalable network solution for the distributed simulation Advanced Concepts Technical Demonstration (ACTD) Synthetic Theater of War (STOW) 97. Prior to Engineering Demonstration-1A (ED-1A), the greatest success for network solutions within the STOW program occurred with the demonstration of the Synthetic Theater of War-Europe (STOW-E) in November 1994, in which 1800 simulation entities were supported on a distributed network with a 1.1-Mbps throughput limitation. The entity count goal to demonstrate the success of RITN in ED-1A was 5000 entities.

Through the course of the demonstration, RITN achieved all of its major objectives:

- A prototype communication architecture was validated that employed multicasting, innovative Application Control Techniques (ACT), and a heterogeneous network using Internet Protocol (IP) on the local area network (LAN), and Asynchronous Transfer Mode (ATM) protocol on the wide area network (WAN), with a bi-level multicast scheme between them.
- The network supported 5249 entities using this prototype scalable architecture.
- Multicasting was introduced to send traffic only to where it was needed to reduce LAN and WAN traffic.
- Implementation of improved ACT reduced network loading and processor demand on individual hosts; it included a bi-level multicast scheme, quiescent entity service (QES), improved relevance filtering, dual-fidelity subscription, and overload management (OM).
- The prototype of a next-generation Distributed Interactive Simulation (DIS) protocol (DIS 3.X) was used to enable simulations to take advantage of the ACT technologies.
- A time synchronization scheme using the Network Time Protocol (NTP) was introduced in conjunction with Global Positioning System (GPS) time servers.
- Application Translator (AT) technology was employed to permit legacy simulations to seamlessly participate in a DIS 3.X exercise.
- IP LANs and ATM high-speed WANs were integrated through a bi-level multicast scheme to provide the backbone bandwidth needed to support a large distributed simulation.
- Network management and data collection techniques were improved using the Simple Network Management Protocol (SNMP) for near real-time data collection.

Among the most significant of findings discussed later in the report are the following:

- Multicasting reduced the volume of traffic received by individual sites by 35% to 60% over that which would have been received with a broadcast delivery scheme.
- The new bi-level multicast and consistency protocols were very robust and show promise for extensibility into future applications.
- The High Performance Application Gateways (HPAG) introduced minimal transmission delays while providing a router interface between the LAN and WAN not available commercially.

- The agent architecture employed was effective and contributed minimal additional traffic to the network.
- The RITN architecture allows for the participation of legacy simulations in the large-scale exercises of the future.
- An exercise of the magnitude envisioned for STOW 97 can be supported by existing backbone technology.
- LAN technologies must be upgraded to support local data loads to host processors.
- The SNMP, with minor adjustments to the ED-1A implementation, will be an effective and indispensable tool for exercise management and rapid data collection in future exercises.

CONTENTS

Appendices

Figures

Tables

1. BACKGROUND

The goal of the Defense Advanced Research Projects Agency's (DARPA) Synthetic Theater of War (STOW) Program is to develop theater-level warfare simulations for training and mission rehearsal at the joint command level as an Advanced Concepts Technical Demonstration (ACTD). The Real-Time Information Transfer and Networking (RITN) program is a component of STOW system program development. The RITN effort grew out of a recognition that STOW needed a scalable network architecture that could expand to handle the ever-increasing demands imposed by larger and larger exercises. Scaling the application to larger theater-sized simulations required development within all the logical layers of the application/network system. Hence, RITN, which combined network and application technology development to evaluate technologies that could provide a scalable network and application solution for the distributed simulation ACTD STOW 97. Further, the network technology evaluation requirements placed RITN in a unique position to determine the required network hardware and software upgrades to the operational simulation network, the Defense Simulation Internet (DSI), which will support STOW 97.

The load on a distributed simulation network grows in proportion to the number of simulated platforms or entities. As exercises increase in size, the simulation load will, at some point, exceed some system capability such as the available bandwidth. To raise this load limit, techniques were developed to increase the density of useful information that can be transmitted across a given simulation network.

The first application of these techniques was in the Synthetic Theater of War-Europe (STOW-E) exercise in November 1994. In this exercise, 1800 entities were to be generated by 18 sites around the world. The network's topography created bottlenecks that would require data traffic at 4.9 megabits per second (Mbps). Unfortunately, the available bandwidth was only 1.1 Mbps; bandwidth-demand reduction techniques (BRT) were required to reduce the bandwidth demand by approximately 80%. Through relevance filtering (i.e., delivering only data required by other sites) and data-concentrating techniques (e.g., efficient protocols, compression, and bundling), this goal was met.

To increase the number of supportable entities even further, these techniques were refined and augmented to improve performance and to reduce or eliminate internal throughput limitations. The Application Control Techniques (ACT) task was launched for this purpose. The ACT task changed the information architecture of the simulation applications to take advantage of newer networking hardware and software/protocol development. The ACT effort was combined with an initiative to employ and advance newer, high-throughput network technologies to form the RITN program. The RITN program's goal was to produce a scalable simulation network to support future large-scale exercises. The first integrated products of RITN were demonstrated in STOW Engineering Demonstration-1A (ED-1A) in November 1995.

2. OBJECTIVES

The RITN program's principal objective is to develop a network solution for the STOW 97 ACTD (November 1997). This network solution must demonstrate scaleability up to 50,000 entities, with a minimum of constraints placed upon the simulation applications. From the network solution, the required technology will be integrated into the operational DSI network.

To achieve this objective, the RITN program is investigating and developing the following technical approaches:

- Multicast technology, which sends data only where needed to reduce utilized bandwidth and loading on the end-user host;
- Bi-level multicast, which facilitates the transport of packets from a dynamic Internet Protocol (IP) multicast local area network to local area network (LAN-to-LAN) service to a less-dynamic Asynchronous Transfer Mode (ATM) wide area network (WAN) service;
- ATM network technology, which provides low-latency, high-speed, and cost-effective communication service;
- ATM multicast with switched virtual circuits, which provides efficient WAN connectivity;
- Quality of Service (QoS) extended from application to ATM service, which ensures low latency and reliability over a shared communication media;
- Integration of Internet Protocol (IP) and ATM service, which include end-to-end IP-based Remote System Verification Program (RSVP) QoS over ATM QoS;
- Route Service, which provides a scalable ATM network;
- FASTLANE, which provides an ATM high-speed network security solution.

As an incremental step toward the principal objective of RITN, Engineering Demonstration-1A (ED-1A) tested the effort's progress. The overall RITN objectives for ED-1A were to:

- Validate a new distributed simulation communication architecture;
- Demonstrate a capability to simulate 5000 entities;
- Demonstrate bandwidth reduction techniques known as Application Control Techniques (ACT);
- Develop a next-generation protocol (DIS 3.X) to take advantage of the new ACTs;
- Develop a bridge to legacy (DIS 2.X) simulations;
- Integrate a bi-level multicasting solution where one level is simulation-application-specific IP and the other is WAN-specific with ATM;
- Advance ATM capability to take advantage of high-speed networks;
- Develop methods to synchronize the system using Network Time Protocol (NTP);
- Improve network management and data collection techniques using Simple Network Management Protocol (SNMP).

To accomplish these objectives, the program defined the following nine areas of investigation:

- Four techniques to reduce the amount of traffic that needed to be sent, received, and processed;
- Two methods to deal with traffic exceeding the system's throughput;
- A way to support legacy simulations in future exercises;
- A method to synchronize all hosts within the infrastructure;
- Implementation of a new, high-throughput network.

To effectively and efficiently manage the volume of simulation data traffic, the following technologies were chosen:

- Multicasting;
- New methods for handling quiescent objects;
- Multicast subscription and agents;
- Multiple fidelity channels.

If these mechanisms were unable to keep the load within the limits of the system's capacity, a combination of overload management and quality service support would ensure that the system was not overwhelmed with data input. To enable many of these techniques, as well as to facilitate post-exercise data analysis, a network timing protocol was introduced to synchronize the clocks of all hosts connected to the system. Previous simulation system architectures did not allow precise time synchronization of processes and, therefore, the ability to correlate simulation events throughout the system was lacking. A new network was also used to provide increased throughput and assess the ATM technology to support distributed simulation. To illustrate the relationship of new simulation technologies, figure 1 describes a network node incorporating each element. The following subsections provide detailed discussions of each network technology developed and tested in ED-1A.

2.1 MULTICASTING

In large distributed simulation exercises, much of the total state of the simulated objects is irrelevant to any particular simulation host. This is generally due to two factors. First, sensors have limits on their maximum range. Simulated objects outside sensor range cannot be detected, so there is no need to transfer their state to the sensor simulation. Second, most military units tend to congregate within close proximity, occupying only a small portion of the battlespace. This tendency, and the fact that most military units are generally simulated on a single host (or a cluster of hosts on a single LAN) limits the state or data about the unit that must be delivered to any other particular application hosts across the network supporting other military units. The RITN/ACT team constructed a prototype relevance filtering approach that attempted to minimize transfer and processing of irrelevant state data. The RITN/ACT system used multicast transport services to implement the prototype relevance filtering scheme. The multicast-based filtering scheme's purpose was to deliver all relevant data to the simulation hosts while reducing the transferred irrelevant data so that network resources and host processing capabilities could be used more efficiently.

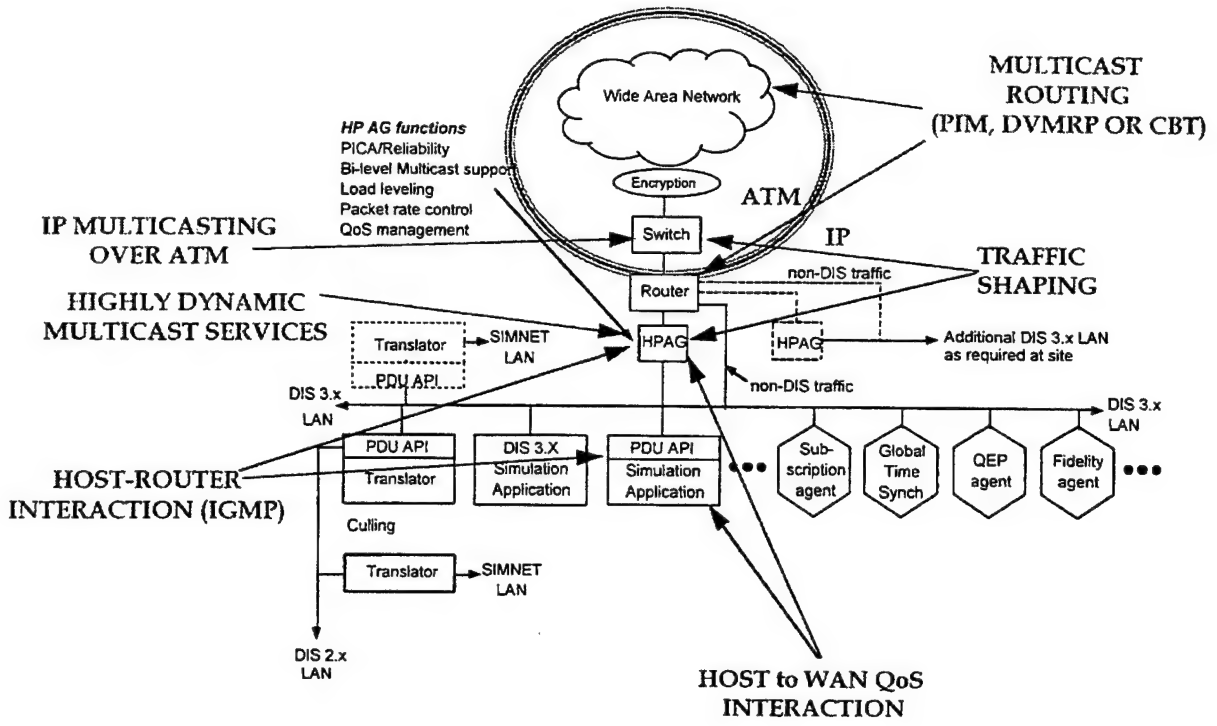


Figure 1. RITN node site architecture.

The multicasting objectives of the ED-1A experiment were as follows:

- Reduce the data traffic on the WAN, tail circuits, site routers, site LANs, and simulation hosts using a prototype multicast-based relevance filter;
- Provide multicast service adequate for the needs of the simulators, with approximately 1000 multicast groups to seven sites and very fast "join times," (a short delay between requesting to receive data from a group and receiving the data sent to that group);
- Evaluate the ability of a bi-level multicast service from a highly dynamic LAN IP multicast to lowly dynamic WAN ATM multicast to determine if the bi-level multicast is adequate for simulator needs, e.g., numbers of multicast groups available and acceptable join/leave times.

2.2 QUALITY OF SERVICE

Though multicasting was intended to reduce bandwidth utilization, there was no inherent guarantee that the packets sent would be delivered. Thus, an additional objective of ED-1A was to investigate providing QoS-based delivery for simulation traffic. QoS is necessary for bandwidth reservation across the network as well as to ensure latency requirements. This is important as networking advances to service-on-demand from network service providers. Since QoS support was not yet present in most commercial products used in the exercise, an architecture compatible with future QoS support was designed.

Based on this overall plan, the objective was to design a scheme to allow feedback to the applications when more traffic was sent than could be handled, and to allow the application some control over which packets would be dropped.

Another objective was to better understand issues of shared reservations, statistical multiplexing of simulation traffic, and aggregation of bandwidth requirements. Later sections discuss QoS issues in more detail.

2.3 QUIESCENT OBJECTS

Current DIS protocols require transmission of states of objects at a minimum rate (the default is 5 sec), even if the state of an object is not changing. This results in transmission and processing of redundant data, inefficiently using system resources and impeding scaleability. Estimates were that 70% or more of the entities in large-scale distributed simulation exercises will be quiescent for much of the time and, hence, that a significant portion of the information created by simulation objects is redundant.

The RITN/ACT effort constructed Consistency Protocol (CP) and quiescent object (QO) detection algorithms. This prototype was intended to reduce redundant traffic due to quiescent entities, i.e., entities that were not changing state. The ED-1A experiment involved the following CP/QO-related objectives:

- Evaluate the effectiveness of the CP/QO prototype at reducing traffic;
- Test whether a Negative Acknowledgment (NACK) based protocol, such as CP, was sufficiently robust in a demanding, real-time network environment subject to high packet loss rates and significant periods of lost connectivity;
- Characterize the numbers of quiescent objects in a large scenario;
- Gather data and insights about how to improve the prototype.

2.4 SUBSCRIPTION AND AGENTS

Simulation system architectures can be broadly categorized as distributed, hierarchical, or centralized. A distributed architecture tends to allocate functionality to individual simulations. This leads to a fairly robust system because the individual simulations are relatively independent and self-sufficient, but this architecture can suffer from scaleability problems because every simulation must support all system functions. In contrast, a centralized architecture tends to allocate functionality to centralized servers. This approach can lack robustness and, additionally, it may not scale particularly well. An intermediate approach is to adopt a hierarchical architecture in which system services are allocated to servers that act on behalf of manageable sets of simulators. This approach can yield both scaleability and robustness. A good example of a successful, relatively scalable and robust hierarchical system is the Internet.

The RITN/ACT effort followed the hierarchical architecture approach by incorporating agents that provide simulation and system services for simulations. The concept of an agent is similar to that of a server. The difference is that an agent enhances the performance/scaleability of a simulation, when present, but is not necessary for the simulation to operate. The RITN/ACT architectural concept supports agents for functions including, but not limited to, subscription, consistency, fidelity, and quality of service.

The agent prototyped by RITN/ACT to try out this architecture was the subscription agent, which provided the following roles:

- Matching subscribers for data with publishers of data;
- Determining the multicast groups needed to permit publishers to send data to subscribers;
- Directing subscribers to join appropriate groups and publishers to send to appropriate groups.

Objectives of the ED-1A experiment for subscription agents were as follows:

- Evaluate the overhead traffic due to the subscription agent approach, including the consistency protocol and discovery protocol;
- Evaluate the robustness of the agent approach in a demanding, real-time network system subject to high packet loss rates, simulator overload, and significant periods of lost connectivity;
- Determine whether the system could operate (albeit at reduced performance levels) in the absence of the prototype agent.

2.5 FIDELITY/UNCERTAINTY TOLERANCE

Supporting simulations of wide-area viewers (WAVs) is a prime challenge in building large and scalable distributed simulation systems. Such simulations (examples include plan-view displays (PVDs) and certain sensors such as radar) are difficult to support because they need to know the state of objects over a wide area of the simulated environment. This characteristic can result in transfer and processing of unacceptably large amounts of traffic if DIS 2.X protocols are used. Fortunately, most simulations that need state from a wide area can also accept a relaxed tolerance for the certainty with which state is known without degrading realism.

Several approaches to providing state updates with a relaxed uncertainty tolerance were proposed and evaluated for prototyping as part of the RITN/ACT effort. These included the following approaches (which are in some cases complementary):

- A down-sampling agent that re-emits reduced-fidelity state updates based on full-fidelity state update rates;
- A fidelity agent that matches uncertainty tolerances of subscribers and publishers and directs publishers to produce one or more channels of the required fidelity;
- Independent or hierarchical multiple fidelity channels;
- Dead reckoning thresholds for controlling uncertainty tolerance;
- Timing management for controlling uncertainty tolerance.

The RITN/ACT effort prototyped a hierarchical, multiple-fidelity-channel approach that used time as a parameter for controlling uncertainty tolerance. Objectives of the ED-1A experiment related to fidelity/uncertainty tolerance included:

- Evaluate and test the effectiveness of the prototype at reducing traffic;
- Gather data and insights showing how to better support simulations of WAVs.

2.6 OVERLOAD MANAGEMENT

Based on experience with previous large-scale distributed simulation exercises, the project personnel were aware that overwhelming the network with data could prove disastrous. Simple solutions to this problem involve dropping packets when the output rate approaches the system throughput

limitation. This prevents overload conditions, but also threatens the validity of an exercise. So, for ED-1A, a more sophisticated approach to Overload Management (OM) was designed, one that would set its response thresholds interactively with the network via the QoS algorithm. This approach would attempt to prevent overload conditions by directing the simulations in its charge to reduce their output level. If this procedure failed to prevent persistent overload conditions, then packets would be dropped as before, but according to a priority scheme whereby frequently repeated Protocol Data Units (PDUs), e.g., entity-state PDUs (ESPDUs), would be dropped in preference to one-time transmissions like Fire and Detonation PDUs. Based on this design, ED-1A established the following OM objectives:

- Evaluate the practicality of setting response thresholds via interaction with the QoS algorithm;
- Attempt to prevent overload conditions by interacting with the data generators themselves via a Source Rate Control (SRC) algorithm;
- Evaluate the impact on exercise validity of priority-based packet dropping via an improved Load Leveling (LL) algorithm.

Neither QoS nor SRC was available for ED-1A; only the third objective was pursued.

2.7 TIME SYNCHRONIZATION

Another ED-1A objective was to provide synchronized clocks to all platforms for use by applications and data collection devices. A related objective was to evaluate the use of synchronized clocks by the principal simulation, Modular Semi-Automated Forces (ModSAF). The implementation of the Network Time Protocol (NTP) allowed simulators to send absolute rather than relative time stamps. Thus, dead reckoning forward from another simulator's state could be based on the time stamp in the packet, rather than on the time of receipt, or the time of receipt minus some assumed propagation delay. This scheme was intended to reduce the effects of jitter (delay variance).

2.8 APPLICATION TRANSLATOR

As the project progresses in its effort to support distributed simulation exercises with new ACTs (eventually evolving into a new DIS protocol tentatively labeled DIS 3.X), support is required to allow the continued participation of a limited number of legacy systems that could be required to meet training goals under particular circumstances. The goal of the Application Translator (AT) effort was to design an intermediary device that would allow legacy systems to interact seamlessly with the more modern components of the RITN system. As implemented, the AT handles the interaction with the various agents and services provided by the ACT system on behalf of the legacy systems it supports. The objective for ED-1A was to demonstrate the viability of using the AT to allow legacy simulators (DIS 2.03 ModSAFs) to effectively participate in a "DIS 3.X" exercise.

2.9 NETWORK ARCHITECTURE AND PROTOCOLS FOR EFFICIENT USE OF BANDWIDTH

The primary objective in the network design for ED-1A was to provide an architecture that would provide the scaleability needed to move from the relatively small number of entities that saturated STOW-E in late 1994 to the large multiple-division level engagements planned for STOW 97. Cost and time were major constraints in considering the technologies to support the selected architecture. Thus, a strong emphasis was placed on the best available commercial implementations of extant or emerging experimental network capabilities. Some initial considerations included the following:

- Use multicast to replace the existing broadcast protocol used in simulation exercises. The offered load to individual simulation hosts in a distributed simulation needed to be limited to only that information required at the workstation for its area of interest in the virtual battle space.
- Continue use of IP-based networks. The primary simulation application used, ModSAF, was IP-based, running on UNIX hosts. Network implementations of multicast were first available in IP networks.
- Use shared 10 Mbps ethernet for LANs at node sites for initial demonstrations needed. Higher speed LAN technologies were not sufficiently mature to introduce in the first demonstration series. Specifically, switched ethernet devices were not multicast aware. Further, it was anticipated that shared ethernet in the LAN would scale sufficiently for ED-1A. To achieve the LAN scaling needed for STOW 97, however, alternatives to shared ethernet will be needed.
- Assess alternative multicast implementations in networked high-speed routers supporting local simulation node sites. The multicast implementations considered were Protocol Independent Multicast (PIM), Distance Vector Multicast Routing Protocol (DVMRP), and Core Based Trees (CBTs). Only two router vendors, Cisco and Bay Networks, using PIM and DVMRP respectively, had both working multicast implementations, and the ability to connect to high-speed ATM backbones.
- Select Open Shortest Path First (OSPF) as the routing protocol since it was a commercial standard available on several routing platforms.
- Adopt ATM as the backbone technology consistent with Department of Defense (DoD) communication policy. ATM offered the potential for bandwidth on demand, and an affordable pricing structure based on actual usage. ATM also offered the potential of very low latency, real-time characteristics, managed quality of service, and offered the future option of high-speed encryption using the FASTLANE encryption system under development. The approach was to treat the WAN as a cloud service provided by the Defense Information Services Agency (DISA) or a commercial provider. Network cloud service is defined by a WAN interconnectivity between distributed site nodes that is transparent between the sites, and is hidden by the WAN service provider.
- Use effective mapping between ATM and IP services. The most important service needed was mapping between IP local area multicast to wide area ATM multicast and back to IP multicast at the distant node site. Since no commercial capability existed for this service, this function was implemented in the High Performance Application Gateway (HPAG) developed as part of the ACT effort.
- Use available high-speed research networks for initial demonstrations.

The design considerations for the node site architecture outlined above are shown in figure 1.

Using the considerations of the network implementation and the overall RITN goal, the network objectives for ED-1A coincide with the multicasting objectives, with the addition of the following:

- Integrate and evaluate IP local area and ATM high-speed WAN technologies that provide the adequate bandwidth, routing, and switching required for large distributed simulations;
- Improve network management and data collection techniques using SNMP.

3. COMPONENTS

This section describes in detail the components of the RITN communication architecture. The architecture evolved from previous experience to support large DIS demonstrations. Prior to ED-1A, the Application Gateway (AG) was developed to improve the scaleability during STOW-E. The AG software resided in a workstation that lay serially between each LAN in the exercise and the WAN. All functionality to scale the STOW-E system lay in the AG. One desire of the ACT design team was to export as much functionality as possible from this single workstation to increase efficiency and capability while reducing the risk introduced by such a single point of failure. Another major goal was to implement a multicasting scheme on both the LANs and the WAN to reduce the delivery of unwanted data packets to every portion of the system, from the WAN backbone to the individual simulation hosts. These two goals implied pushing the BRTs down as far as possible in the simulation chain, even to the simulation applications. As a result, it became necessary to provide an additional mechanism to permit legacy simulation hosts to participate in this new architecture. The sum of these requirements led to a four-element physical architecture to support each simulation LAN with multiple algorithms spread among them.

The RITN system can, thus, be dissected along two lines. One is by physical component, the other is by functionality or algorithm. To the user community, the logical breakdown is by physical component, since the functionality of the system is transparent to the user. This physical description of the system is illustrated in figure 2.

To really understand the interactions that result in the RITN system's performance, however, it is also necessary to understand the breakdown and interrelationships of the algorithms. For this reason, each functionality is also described below, and the data flow is illustrated in figure 3.

3.1 HIGH PERFORMANCE APPLICATION GATEWAY

The first element of the ACT architecture is labeled the High Performance Application Gateway (HPAG). This workstation is positioned between the LAN and WAN in the same manner as the STOW-E AG, but performs a much shorter list of functions to reduce processing latency. The HPAG is responsible for implementing the bi-level multicast scheme (supporting a larger number of multicast groups on the LAN than on the WAN), dynamically interacting with the network to assure QoS, and providing a throughput safety valve in an OM function (linked to the QoS function) capable of initiating both LL and SRC. As the WAN did not support dynamic QoS, this software was not implemented. SRC was not yet incorporated into the ModSAF simulation software, so the OM software, with no QoS code to link to, was reduced to a static load-leveling filter that was never invoked.

Bi-level multicast is a technique for implementing IP multicast service conforming to the standard IP multicast service model ([RFC1112]). Bi-level multicast is implemented by several interdependent subfunctions:

1. The HPAG queries the LAN to determine what groups are joined locally.
2. The HPAG reliably communicates this list of locally joined groups to all other HPAGs.
3. The HPAG joins an appropriate subset of the available 127 WAN groups (an optimum number calculated from the number of LANs involved in the exercise) that encompasses the required LAN groups.

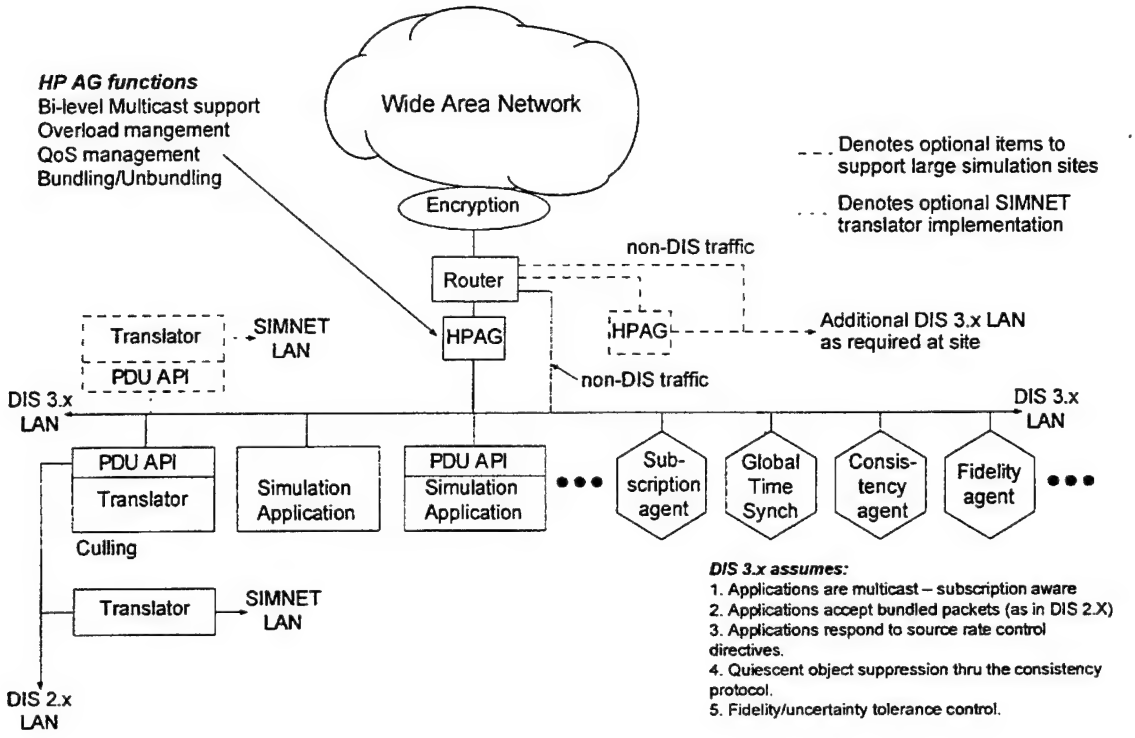


Figure 2. The physical description of the RITN system.

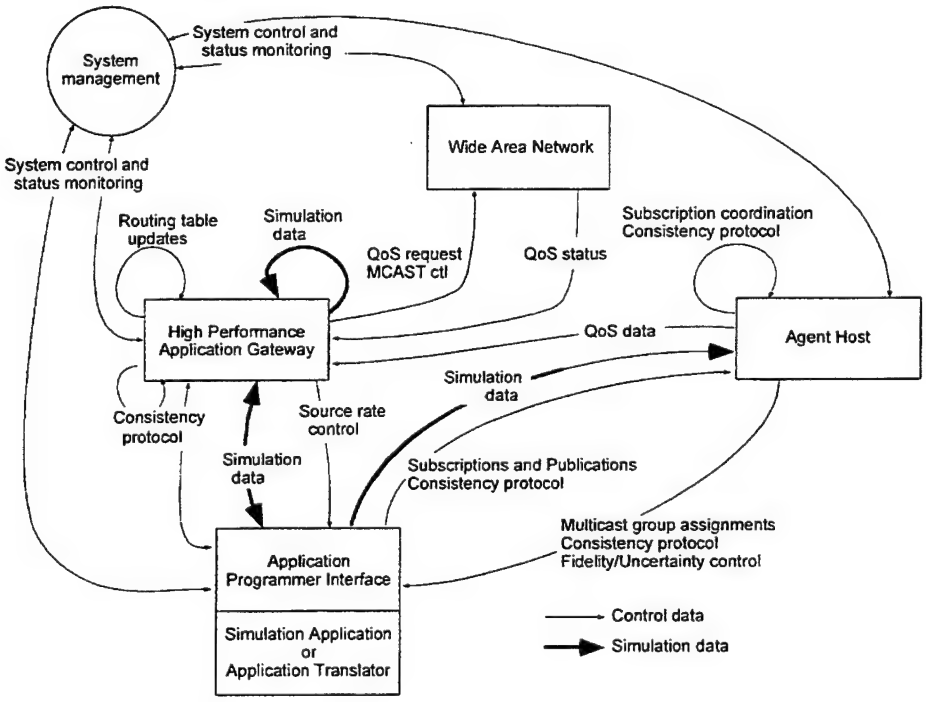


Figure 3. The data flow of the RITN system components.

4. On receiving a data packet from the LAN, the HPAG computes that other sites need the packet, based on information received in item 2. The HPAG then encapsulates and forwards the datagram over the appropriate WAN group, causing it to be delivered to the appropriate set of sites.
5. On receiving an encapsulated data packet over the WAN, the HPAG checks to see if the packet is needed locally. If so, it is delivered to the LAN. If not, it is dropped.

With this scheme, multicast data packets are sent only to sites that need them. This is intended to reduce WAN loading and also the incoming processing load at each site. It is also intended to reduce LAN loading at each site, since only needed traffic appears.

The other ACT located in the HPAG is OM. OM is a scheme to cope with overload conditions in a more intelligent manner than letting the underlying network drop packets in an arbitrary manner.

OM serves to limit the amount of data transmitted over the WAN, and to constrain the volume of data to the boundaries of the resource reservations determined by the QoS algorithm. The goal of the OM algorithms is to curtail excess data generation as close as possible to the data source.

The design for OM processing has two distinct subelements that control the long-term behavior of the simulation applications. The first, SRC, attempts to ensure that traffic generation does not exceed the capacity of the WAN. Load Leveling (LL), on the other hand, drops packets based on priority if overload conditions persist (priority is encoded in the IP type of service (TOS) field). Thus, if packets must be dropped, packets with lower priority will be dropped preferentially. Simulators use this scheme by labeling packets with appropriate priorities so that repetitive ESPDUs are dropped before single-transmission Fire or Detonation PDUs, for example. As an overload condition approaches, the SRC algorithm, as directed by the LL algorithm, attempts to reduce bandwidth utilization by reducing the output rate of individual simulations on overloaded network segments. These output rate reductions are intended to maintain WAN utilization rates within the resource reservation limits established in coordination with the QoS algorithm. As neither SRC nor QoS was implemented for ED-1A, however, the only capability supported by the implemented OM algorithm was dropping packets based on packet priority when an overload condition existed.

3.2 APPLICATION PROGRAMMING INTERFACE

The second element of the ACT architecture is known as the Application Programming Interface (API). The API is not necessarily a separate component; it is additional code within ModSAF and other applications that support multicasting. It is also a form of QO service discussed previously. Multicasting permits outgoing packets to be addressed so that they are routed to interested recipients alone. The QO service, on the other hand, enables the simulation of quiescent objects with a vastly reduced number of transmitted packets while increasing (over STOW-E values) the number of entities that can be treated as quiescent.

The API developed for the RITN program provides the primary interface between simulation applications (ModSAF simulations, stealth viewers, PVDs, and the AT) and the underlying ACT components. This is not an API in the usual sense—it does not provide a single isolation layer between the application and the ACT components. Rather, "the API" was the name given to a collection of modules that tie application modules to the underlying ACT infrastructure.

Three key technologies are embodied in the API:

- Use of IP multicast on commercial off-the-shelf (COTS) hardware using COTS operating systems (UNIX);
- A clean and efficient separation of message content from network representation (a true presentation layer abstraction);
- An interface to link legacy software modules to new network transport mechanisms with a minimum impact on application modules.

3.2.1 IP Multicast

IP multicast is an experimental extension to the Internet Protocol Version Four (IPV4) protocol stack. A key question was whether the implementations of IP multicast offered by commercial workstation vendors were mature enough for real-world applications. In general, the implementations were lacking, but it was possible to work around problems to reach operable solutions.

Hundreds of multicast groups were successfully used on Silicon Graphics, Inc. (SGI), Sun, and Hewlett-Packard (HP) workstations. In every case, however, some kind of work-around was necessary. On the SGI and the HP, only 20 addresses per socket could be subscribed to, so multiple sockets had to be opened to make all the subscriptions. Since multicast subscriptions are per machine, however, and happen at the IP (not the User Data Protocol (UDP)) layer, these sockets did not need to be bound to a particular address. It thus created a bookkeeping problem and used up precious file descriptors, though it did not present a performance problem.

The same process of opening up more sockets could have been used on the Sun workstations, but a call to Sun resulted in an increase in the constant of addresses per socket from 20 to "a very large number." With the kernel patch that Sun now delivers (and will include in their next operating system (O/S)), nothing out of the ordinary is required. SGI has agreed to do the same thing with their next O/S.

Unfortunately, every vendor uses an inadequate implementation of multicast filtering in their kernels and, thus, the performance drops precipitously after subscription to the first 20 addresses. SGI has promised to fix this in a future release of their O/S. As it stands today, when more than 250 multicast addresses are subscribed to, better performance can be realized by implementing multicast in application space, and by using a promiscuous socket (though this is not envisioned).

Finally, it appears that none of the kernel implementations will ever be anywhere near as efficient as hardware filtering by the ethernet devices. Thus, it would be worthwhile to take the extra effort to pick IP multicast addresses that will tend to avoid collisions better when translated to ethernet, and then hash buckets. The odds are good that all the vendors use the same hash, although this is not known for sure.

3.2.2 Presentation Layer Abstraction

Another key question addressed by the API is whether the content of DIS PDUs can be separated effectively from their network representations. This has become a significant issue because the network representations of DIS data are becoming far more complex than in the past.

A software module called the PDU API was developed to abstract network bindings away from the application. This module uses data files rather than code to define protocol structures. By doing

so, it allows far more sophisticated structures to be used to arrange information without the need for custom programming for each PDU. The primary concern that prevented the creation of such a module in the past was performance. It was expected that the interpretation and copying of network data had a negative impact on software efficiency. However, the PDU API avoids this problem by making the bulk of the interpretations ahead of time, using a pattern-matching function. This function generates a set of rules that are processed very quickly and efficiently to transfer network data into application formats, and vice versa.

The PDU API module was successfully integrated into the Simulation, Training, and Instrumentation Command (STRICOM) ModSAF 2.0 baseline, and is now used by over 100 ModSAF customers. The implementation had no measurable impact on performance.

3.2.3 Interfacing Legacy Modules

The final task under API was the integration of new ACT components into the ModSAF software base. Primarily, this consisted of the Consistency Protocol, which is used for efficient communication of quiescent entities, and the Subscription Principal, which coordinates multicast addressing with the Subscription Agent (which runs on the Agent Host). One of the challenges involved in this work is to integrate these mechanisms with a minimum impact on the existing ModSAF baseline software.

The integration approach used for the CP was to modify the one ModSAF module (libEntity) that is responsible for communicating information about entities via DIS. By isolating these changes completely within this module, none of the other ModSAF modules are aware of the CP.

The integration approach used for the Subscription Principal was to design a separate interface module that occupies a layer at the top of the ModSAF software hierarchy. In this way, the module could probe through the ModSAF modules and deduce the need for subscriptions and publications. Then it calls the Subscription Principal, as needed. By structuring the software this way, the existing ModSAF modules no modification to support subscription.

All of these changes were made available to the ModSAF community by inclusion in the 2.0 ModSAF release. However, since these are not standard DIS protocols, they are disabled by default. A set of defects identified during ED-1A testing will be integrated into the ModSAF baseline for inclusion in the ModSAF 2.1 release.

3.3 AGENT HOST

The RITN/ACT architecture's third element, the Agent Host (AH), is an execution platform for agents. Agents are software modules that perform functions that enhance the performance of individual simulations and the system as a whole. The portion of a simulation or other system component that interacts with an agent is termed a principal. Agents and principals in the STOW/RITN architecture have a relationship similar to that of agents and principals in a commercial transaction, i.e., agents perform tasks on behalf of principals that are perhaps too onerous, or require arcane skills or knowledge that the principals might not possess.

Agents are different from servers in that they are not required for the system to operate. If a particular agent is unavailable, performance may suffer and individual simulations may be overloaded, but no functionality is lost. A key concept related to agents is that they are software modules that may (from an architectural point of view) be migrated to execute at various places in the system in

order to optimize a particular exercise's performance and resource tradeoffs. STOW RITN/ACT constructed the Agent Host as a convenient platform for prototyping and experimenting with the agent concept.

Of the several agents identified in the STOW RITN/ACT architecture (subscription, consistency, fidelity, quality of service, etc.), only the subscription agent and its principal were fully constructed. The subscription agent performs the following tasks on behalf of its principals:

- Matching of data subscriptions and publications;
- Resubscription/republication;
- Multicast group assignment;
- Fidelity channel control.

The subscription agents accept data subscriptions and publications from their principals. Subscriptions consist of specifications of the types and ranges of data in which a simulation is interested (a region of interest). Publications consist of specifications of the types of data generated, including rate and size information useful for constructing quality-of-service requests. The subscription agents use a simple two-dimensional grid approach to match publications and subscriptions. Multicast groups are assigned to groups to enable communication between publishers and subscribers. Subscribers are informed of the groups to join, and publishers are informed of which groups are to receive their transmissions.

The simple group assignment algorithm (i.e., grids) prototyped in the initial implementation of the subscription agent is fully distributed to the subscription agents, i.e., no interaction between agents is required. This is possible because all subscription agents have a consistent view of the grid parameters, i.e., cell size, origin, etc. This approach may be viewed as a simple and expedient approach to data clustering. As both simple and expedient, it is also suboptimal. Future implementations of the subscription agent will support more sophisticated data-clustering approaches. These will certainly require exchange of publication and subscription information between subscription agents to achieve a more optimal routing of traffic through the network.

The AH is implemented in C++ using the Modular Tactical Gateway (MTG) software framework, also used in the AT and HPAG components. An AH running a subscription agent was fielded at each of the network sites.

3.4 APPLICATION TRANSLATOR

The final element of the ACT architecture is the Application Translator (AT). The AT enables the participation of legacy simulations (designated "DIS 2.X simulations") by performing all of the tasks accomplished for the so-called DIS 3.X ModSAFs by the API. The AT stands as a peer on the normal, DIS 3.X LAN, and communicated to a second, DIS 2.X LAN, through a second ethernet interface in much the same way that the HPAG stands between the DIS 3.X LAN and the WAN. (Only one DIS 2.X LAN and AT were used in ED-1A.)

Functionally, the AT possesses two data paths: Packets received from the DIS 3.X LAN are processed and passed out to the legacy LAN. Packets received from the legacy LAN are processed and passed out to the DIS 3.X LAN. The AT provides the following services for packets destined for the DIS 3.X LAN:

- Subscription (multicast/relevance filtering);
- Quiescent entity determination (QED) of legacy entities;
- PDU culling (dropping of packets that did not need to be sent out, such as local collision PDUs).

For packets destined for the legacy LAN, the AT provides the following services:

- Entity regeneration, expiration checking;
- QED of DIS 3.X entities;
- PDU culling.

Requests by the AT to the AH on behalf of the entities generated by the legacy LAN are made in support of the consistency, discovery, subscription, and QO protocols.

3.5 NETWORK IMPLEMENTATION

Several DoD and Defense Advanced Research Products Agency (DARPA) high-speed experimental networks offered substantial resources at very little cost to the STOW effort, provided a distributed test environment over extended distances, and had the potential to connect primary development facilities contributing to the STOW effort. The primary networks used in the extended RITN test bed to support STOW were the Advanced Communication Technology Satellite (ACTS) ATM Internetwork (AAI) and the Advanced Technology Demonstration Network (ATDnet), as shown in figure 4.

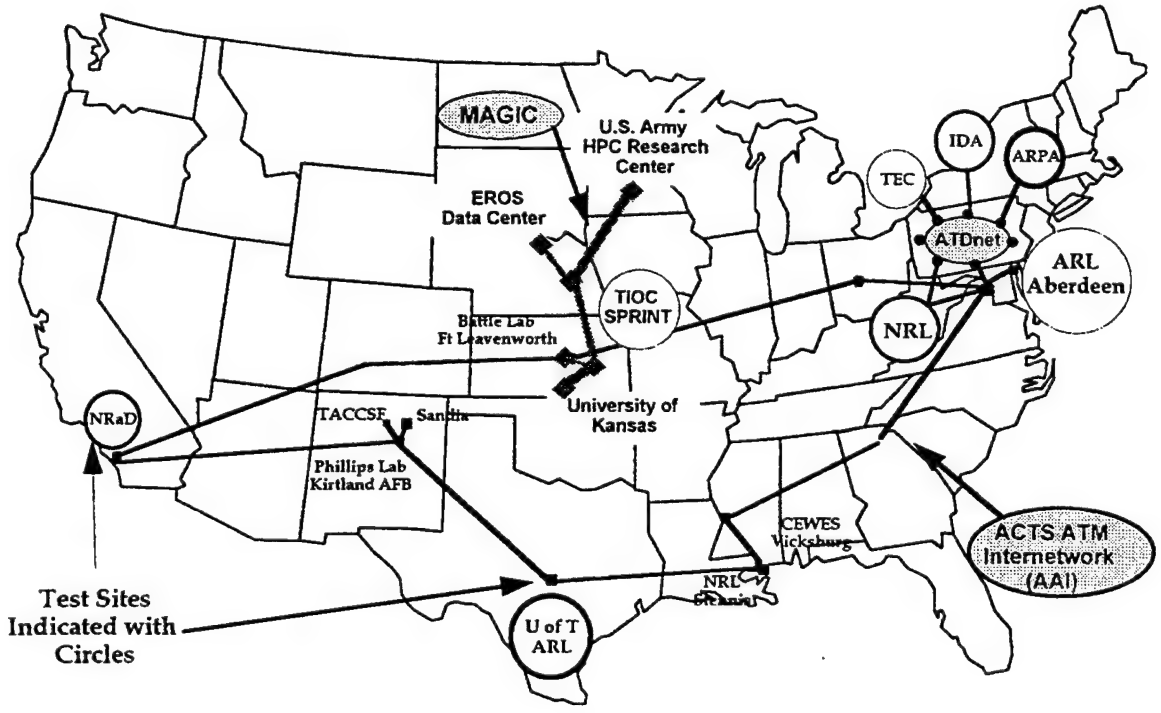


Figure 4. Research networks used in RITN test bed.

1. The AAI is primarily intended to provide a network test environment in conjunction with the ACTS experimental communications satellite. Another focus of the AAI is ATM research, which complements the STOW need for high-speed backbones to support distributed simulation. The AAI is based on a commercial Sprint "cloud" service that is transparent to the edge users. AAI service is DS-3 (45 Mbps). The AAI is a coast-to-coast Continental United States (CONUS) network connecting DoD high-performance computing (HPC) facilities as well as RITN sites.

2. The ATDnet (see figure 5) is a Washington, DC, metropolitan area high-speed experimental network with backbone capacity of OC-48 (2.4 Gbps). Node site connections to ATDnet are OC-3 (155 Mbps). Several RITN node sites are already on ATDnet, and others in the metropolitan area can be added at relatively low cost. The ATDnet is a private, experimental network that can conduct controlled R&D. The primary ATM edge switches in ATDnet are the Fore ASX-100 and the ASX-200 series. Within ATDnet, Switched Virtual Circuit (SVC) signaling is accomplished by both UNI 3.0 and Fore SPANS signaling.

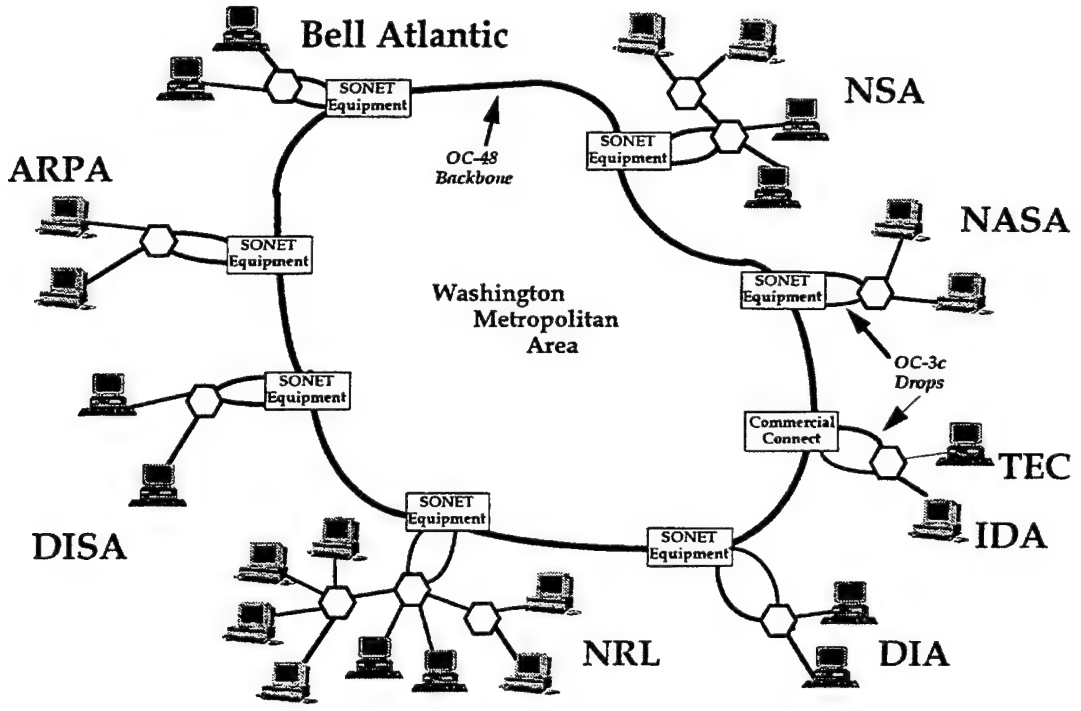


Figure 5. ATDnet in Washington, DC area.

3. ATDnet and AAI bridging. The Naval Research Laboratory (NRL) is directly connected on both the AAI and ATDnet networks and provided the bridging between the two networks for the RITN test bed. Although SVC service is possible across AAI/ATDnet, the ATM protocols do not support the number of multicast groups needed for distributed simulation with SVC signaling. The inability to use the automated features of SVC signaling increases the management burden, requiring manual setup of permanent circuits across the RITN network. Using permanent virtual paths (PVPs) and permanent virtual circuits (PVCs) within those paths was feasible for the relatively small RITN test bed of six sites. Scaling to a larger network without reliable SVC signaling would be unmanageable.

3.5.1 ED-1A Sites

The sites chosen for initial connection in the RITN test bed reflect primary participants in the STOW development effort and the geographic diversity representative of a prototype STOW environment. Originally, six sites were selected. Six sites were viewed as the minimum needed to provide a rich enough topology to effectively test and assess multicast and other technologies that offered the potential capability to scale to STOW 97-level exercises. The initial sites included:

1. Naval Research Laboratory (NRL), Washington, DC. NRL offered existing infrastructure and connectivity to both the AAI and ATDnet networks. NRL is a national leader in ATM R&D and network research.
2. Naval Command, Control and Ocean Surveillance Center RDT&E Division (NRaD), San Diego, CA. NRaD was the primary systems integrator for the successful STOW-E exercises. NRaD was tasked with follow-on implementation of Application Control Techniques for ED-1A and systems integration, test management, and development of the Navy and Marine portions of the synthetic forces (SF) to be used for STOW 97.
3. Institute for Defense Analyses (IDA), Arlington, VA. IDA was a key player in STOW-E and maintains a comprehensive simulation center. IDA is active in the Distributed Interactive Simulation (DIS) community, with extensive experience and resources for simulation exercise participation and application development.
4. U.S. Army Topographic Engineering Center (TEC), Fort Belvoir, VA. TEC is responsible for developing the environment for STOW 97, including terrain, weather, and other environmental features.
5. Applied Research Lab, University of Texas (ARL-UT), Austin, TX. ARL-UT is tasked with development of the primary simulation application for STOW 97, ModSAF.
6. Defense Advanced Research Projects Agency (DARPA), Arlington VA. DARPA is the overall program sponsor. DARPA was already connected to the ATDnet and has a sophisticated demonstration facility, the Enterprise Room, that allows participation in the simulation exercises and evaluation of network performance.

Initial implementation of the RITN test bed at the six sites outlined above is illustrated in figure 6.

The test bed configuration used for ED-1A was modified somewhat from the initial six-site test bed. As reflected in figure 6, a heterogeneous mix of Cisco 7000 and Bay Networks 72000 routers were used initially. The first series of Sub-System Integration Tests (SSIT) using a mix of Cisco and Bay routers was successful. As test loading increased, however, the test bed became increasingly unstable. Several factors resulted in modifying the RITN test bed for ED-1A, as reflected in figure 7. Issues forcing the change in the architecture included:

1. The Cisco routers failed under moderate multicast loads. All sites were then populated with Bay Networks routers that could handle the traffic loads and protocols required for ED-1A.
2. The resources at NRaD exceeded what could be effectively run on a single LAN. Too many traffic generators on a single LAN would overload each other as each host attempted to screen all traffic for relevance on the LAN. Also, the number of hosts available would have saturated a single ethernet LAN. Thus, NRaD was organized as two node sites behind a single router.

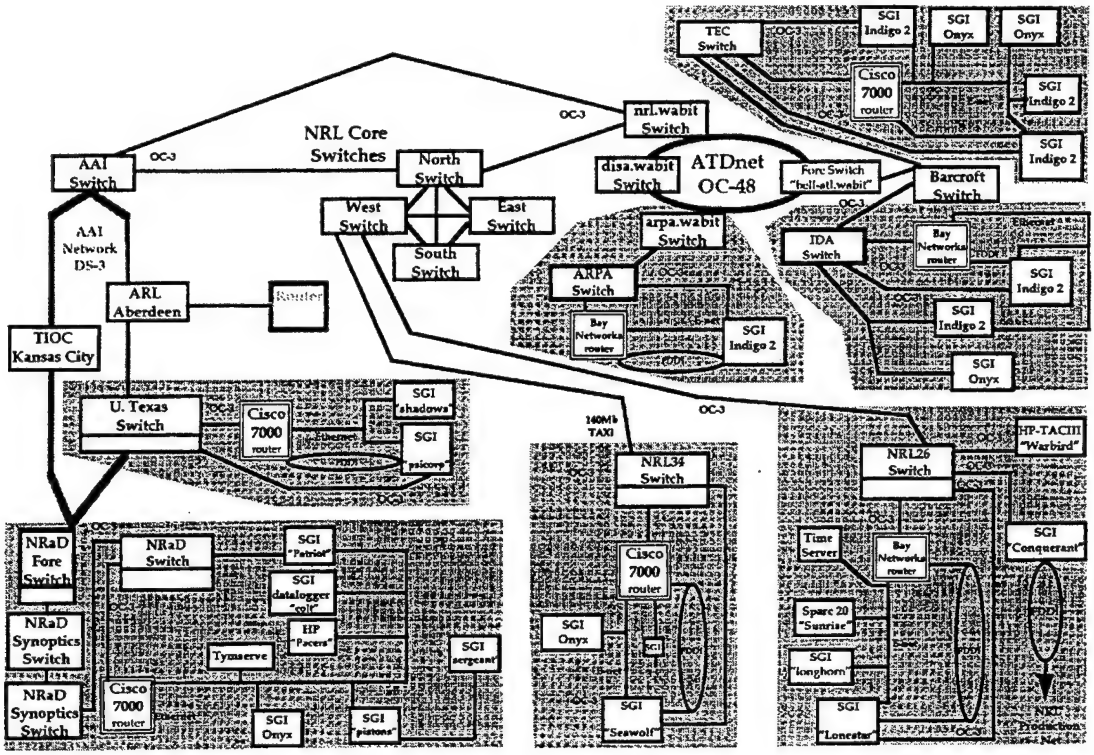


Figure 6. Initial six RITN test bed sites.

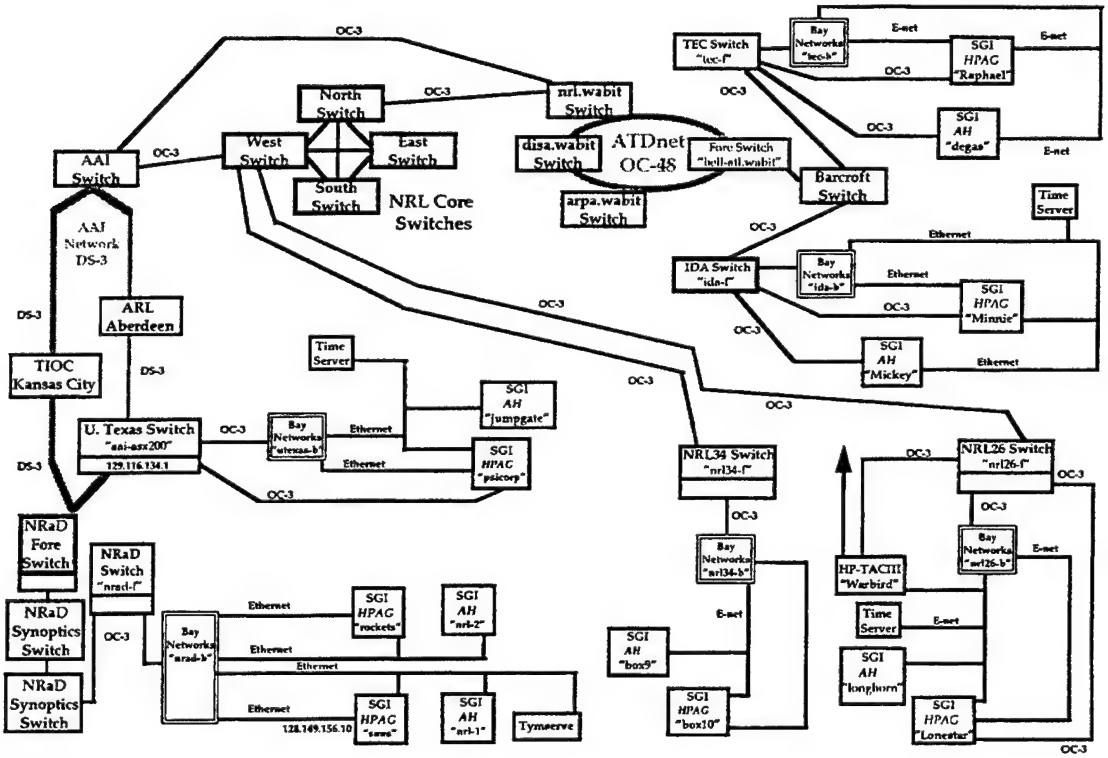


Figure 7. ED-1A network test configuration.

3. The routers were connected by a full mesh of ATM PVCs in the high-speed backbone, with the routers communicating using OSPF group mode over the PVCs. This design choice was relatively easy to manage with one logical IP class "C" subnet for routing, and 30 PVCs making up the connection mesh in the ATM high-speed backbone. This design, however, proved very fragile because of the unreliable nature of the underlying ATM network. Individual PVCs or total ATM connectivity to an individual site was lost. The network became partially segmented, and the routers began competing with each other to assume designated router status for OSPF routing. Even moderate-level exercises, such as ED-1A, could not be supported in this environment. A more complicated scheme was very successful. OSPF group mode was replaced by direct mode. With direct mode, each PVC in the ATM mesh had a unique IP subnet assigned (the original class "C" was segmented for this purpose). The result was that the routers would route around down PVCs, and maintain a coherent OSPF picture as long as a minimum set of VCs were available. There were several occasions during ED-1A (figure 8) where up to half of the RITN PVCs were down, but full node site connectivity was maintained. The penalties incurred were several milliseconds additional delay for each additional router hop and additional router loading (not an issue for ED-1A, as the network was substantially over provisioned).

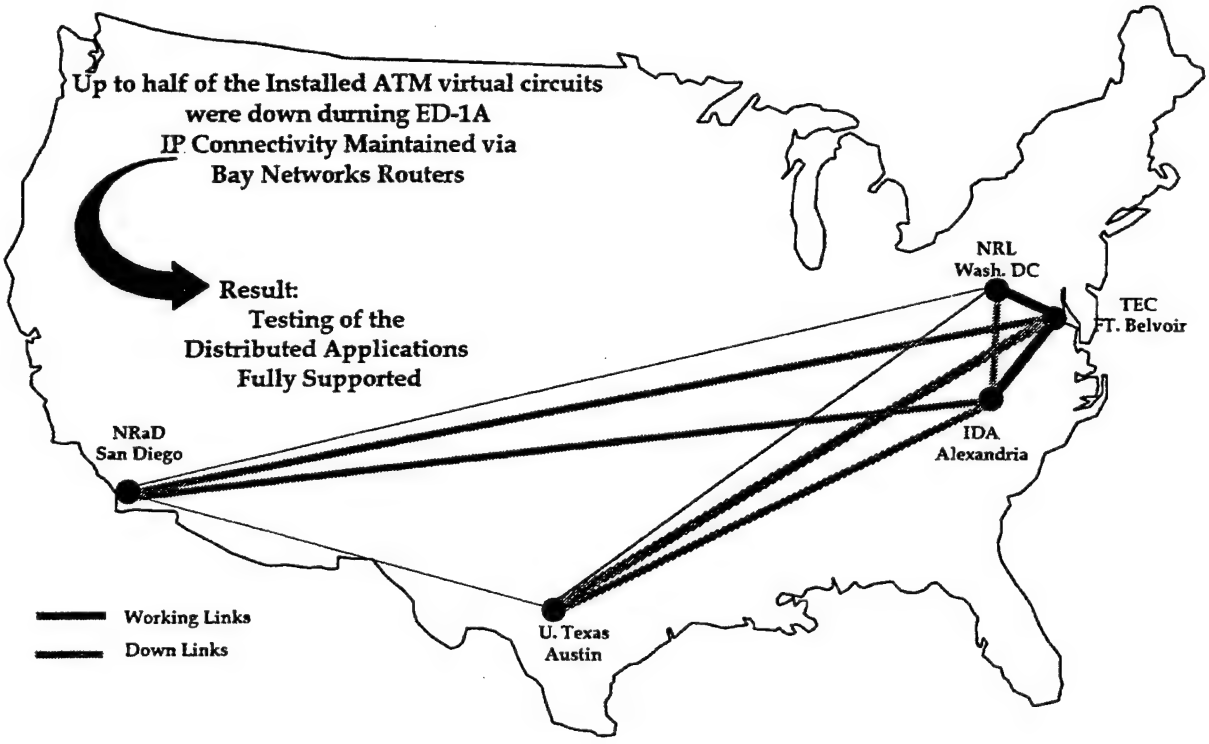


Figure 8. OSPF routing around down PVCs.

3.5.2 Node Site Components

The actual components for a node site are shown for NRL in figure 9. The other node sites were similarly structured. NRaD and IDA had about twice the number of simulation hosts as shown for NRL. TEC had about the same number as NRL, and Texas had fewer (four ModSAF simulation hosts).

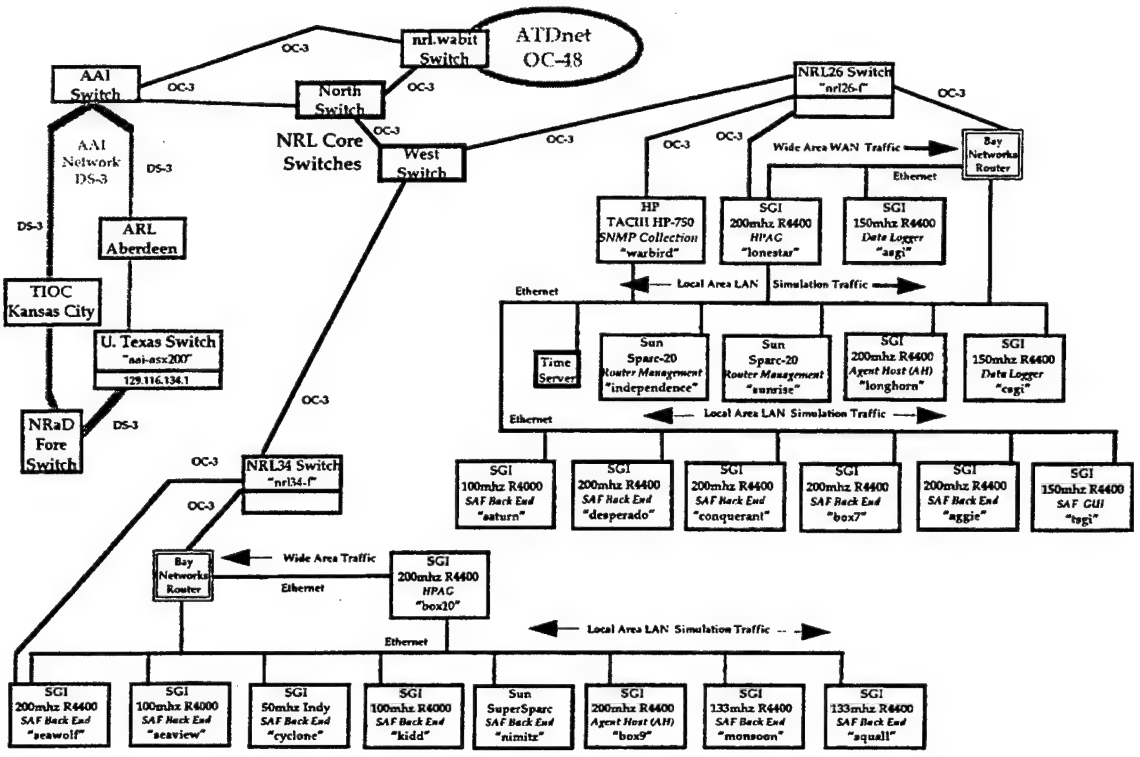


Figure 9. Node site configurations at NRL34 and NRL26.

3.5.3 ED-1A Logical Network

Figure 10 outlines a simplified view of the ED-1A logical architecture and summarizes the details presented above. The key elements in the architecture are:

1. An ATM "cloud" service is provided for the high-speed, wide-area backbone. As shown in figure 10, the AAI and ATDnet are bridged as elements within the cloud that are transparent to the user.
2. An OC-3 (155 Mbps) ATM connection provides connectivity from the cloud to the node site routers. Each router has a PVC connection to every other router, providing a full-VC mesh in the ATM backbone.
3. The routers provide OSPF and multicast routing services.
4. Each router supports one or more IP LANs running distributed interactive simulation applications.
5. Application services are provided on each of the LANs for IP over ATM multicasting, subscription to multicast groups, time synchronization, and other functions.

3.6 ACT PROTOCOLS AND ALGORITHMS

This section summarizes the notable characteristics of the defining protocols and algorithms developed by the STOW RITN program and employed during the ED-1A experiment. This section provides some background information on how the different bandwidth-reduction techniques were implemented in the ED-1A system.

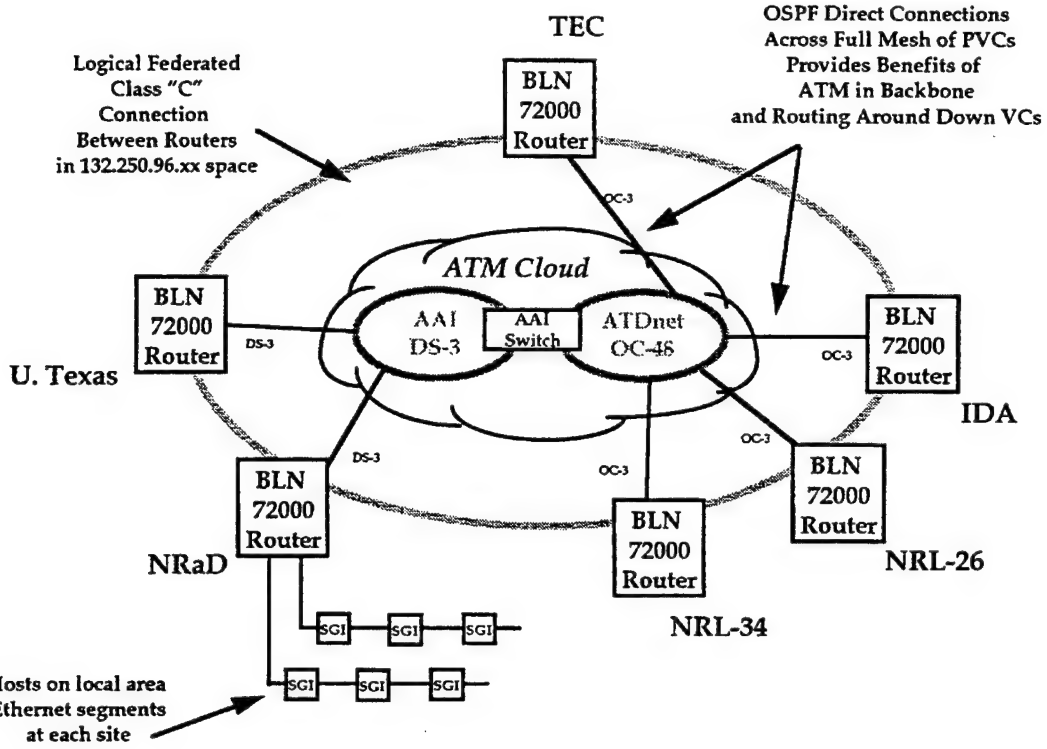


Figure 10. ED-1A logical network.

3.6.1 Multicast Group Assignment/Relevance Filtering

Relevance filtering was implemented using multicast data transmission. Multicast groups were associated with gridded regions or grid square segments of the two-dimensional terrain surface. Simulations joined groups associated with grid squares that were of interest (i.e., fell within the collective sensor range of its entities) and transmitted data for each entity to the group associated with the grid square segments in which that entity was located. The network relayed data from transmitters to receivers using a multicast transmission so that each simulation application received only the data sent to the groups that were joined. This approach significantly reduced traffic received by any one simulation application, especially for large scenarios.

A uniform array of square, two-dimensional grid cells was used to associate groups with grid segments. A multicast group address was associated with each square grid segment. Regions of interest were represented as rectangular regions that were mapped onto the grid array. Each simulation joined the groups for all grid segments that were overlapped, either fully or partially, by a region of interest. State updates were transmitted to the group associated with the grid segment where each entity was located. As described in the following section, multiple grid systems were employed to support a fidelity channel scheme.

3.6.2 Fidelity/Uncertainty Channels

To support WAVs such as PVDs and radar, state data were made available at two levels of fidelity or uncertainty. These two levels were designated by two fidelity channels: high-fidelity and low-fidelity. The high-fidelity channel was designed to supply updates to applications with less tolerance for state uncertainty or error, while the low-fidelity channel was designed to supply updates to

applications with relaxed tolerance for state uncertainty. Simulations subscribed to high-fidelity data or low-fidelity data streams through the subscription mechanism supported by the agent host.

The fidelity channels had a hierarchical relationship in that the low-fidelity stream of data was a subset of the high-fidelity stream. In other words, a subscriber to the high-fidelity stream receives all the updates generated by an entity, while a subscriber to the low-fidelity channel receives a subset of those updates. No extra updates are generated. This approach minimizes the redundant data that might have been transmitted in supporting a nonhierarchical fidelity channel scheme.

The fidelity channels were implemented through multicast transmission, with time as a parameter for controlling the update rates, and uncertainty on each channel. Each region of the terrain had two multicast groups associated with it, one from each of two grid cell arrays, as described in the previous section. The two grids were designated the high-rate grid (containing the high-rate groups) and the low-rate grid (containing the low-rate groups). To subscribe to a high-fidelity channel for a particular terrain region, a simulation joined both groups. To subscribe to a low-fidelity channel for a particular terrain region, a simulation joined only the low-rate group. The time since the last update was used as a parameter to control the destination group to which each update was sent. The algorithm works in the following manner. If the time since the last low-rate group update exceeded a threshold, a pending update was sent to the low-rate group; otherwise, such an update was sent to the high-rate group. The threshold was set to ensure that subscribers to low-fidelity channels did not time out and delete entities that were transmitting only to a low-fidelity channel.

3.6.3 Consistency Protocol

The important feature of the Consistency Protocol (CP) was the data item. Data items are subscribed for, created, updated, or deleted. The state of each data item is kept consistent by the CP throughout the system. The CP is a Negative-ACKnowledgment-based (NACK-based) protocol that employs NACK suppression to enhance scaleability. The protocol has the following characteristics:

- A sequence number is associated with each data item so that each state change of a data item may be differentiated. The sequence number is updated each time the state of a data item changes.
- Data items have only a single owner, but may have multiple readers. Only the owner may modify the state of a data item.
- Hosts transmit periodic messages at a consistent rate or heartbeat, containing the latest sequence numbers for the data items it owns. Readers use the contents of these heartbeats to check whether they have a consistent state for their locally cached data items.
- Upon detecting an inconsistency, readers transmit a request (a NACK) for retransmission of the needed state. NACKs are not sent immediately. Instead, transmission time is randomly selected from a backoff window, and the NACK transmission is scheduled for that time. If a NACK for the same data is received from another host, a pending NACK is suppressed. Also, NACKs are suppressed by receipt of the necessary data, which may have been sent in response to another reader's NACK.
- Owners retransmit requested data in response to receiving a NACK, but do so only after a retransmission window time has expired. This optimization reduces excessive retransmission of redundant data.

The CP PDUs are of the following types:

- Update. Conveys changes to the state of a data item.
- Full. Conveys the full state of a data item. Also used to create a data item.
- Delete. Deletes a data item.
- Request. Requests retransmission, i.e., a NACK.
- Refresh. A heartbeat containing the sequence numbers of the data items owned by a host.
- Group request. Requests all data items in a group. Issued upon joining a group.

The CP is supported by an American National Standards Institute (ANSI) C library, libcp, which provides an API for employing and controlling the protocol.

3.6.4 Quiescent Entity Suppression

To address the issue of entities emitting updates at the default minimum rate when entities are not changing state, the STOW RITN program developed algorithms for detecting when entities have become quiescent. When an entity is deemed quiescent (three consecutive timeout state updates with turret dead reckoning), a final update is transmitted. No more updates are sent, the updates are suppressed until the entity becomes active again, or a request for retransmission of the entity's state is received via the Consistency Protocol. The final update includes an indication that the entity is quiescent. Receivers of a final update note that the designated entity is quiescent. Quiescent entities are not dropped from the simulation exercise as active entities would be if no traffic was received.

A key aspect of the quiescent entity detection (QED) algorithms is an implementation of turret dead reckoning. This approach enables entities that are slewing their turrets (a common modeled behavior), but otherwise not changing their position in the virtual world, to be declared quiescent and subject to suppression of their updates.

To ensure state consistency in the face of lost data or group joins and leaves, the CP is employed. The quiescent entities for each host were maintained on consistency list data structures, which are maintained in a consistent state by all relevant receivers through CP mechanisms. The consistency list is implemented as a CP data item, as outlined in the previous section. Simulations use the information on the consistency list to ensure they have a consistent view of the state of the relevant quiescent entities simulated by each host.

The quiescent entity suppression technique reduces network traffic by eliminating the per-entity periodic heartbeat messages. Effectively, the per-entity heartbeats were replaced with per-simulation host CP heartbeats and related CP protocol traffic.

Throughout this report, the abbreviations QES and QO are used interchangeably to refer to the approach described in this section.

3.6.5 Discovery Protocol

The Discovery Protocol (DP) provides services in which applications can discover agents or other resources available in the system, and monitor their status. The protocol contains only a single PDU (the advertisement PDU) that is periodically emitted by simulators, the AH, etc. An API implemented as an ANSI C library, libdp, permits applications to employ and control the protocol.

The advertisement PDU contains information about the identity and capabilities of the sender. In addition, it contains sufficient information to permit receivers of advertisements to contact the sender in order to utilize the advertised services.

4. TEST METHODOLOGY/PROCEDURES

This section outlines experimental architecture, data collection methods, and experimental design for background purposes. It then describes the methods used to test the effectiveness of each ACT.

4.1 EXPERIMENT ARCHITECTURE

ED-1A employed an ATM WAN (a union of the ACTS AAI and the ATDNet) to link seven LANs at five sites. The WAN passed through four switches and interfaced with the LANs via Bay Networks routers. Two LANs shared a single router at NRaD in San Diego, CA. An additional two LANs (each served by its own router) were set up at NRL in Washington, DC. Three single LANs were served by individual routers at IDA in Alexandria, VA; TEC in Fort Belvoir, VA; and ARL at the University of Texas in Austin. All LANs supported SGI workstations connected by ethernet and running ModSAF to generate DIS entities. SGI workstations also acted as data loggers at NRaD, NRL, and IDA. Two HP workstations were used to manage real-time data collection, and Sun workstations were employed on the last day of the exercise to generate additional ModSAF entities. Approximately 62 ModSAF back-ends were employed to generate entities for the large scenarios investigated (3000 to over 5000 entities), while fewer were used to support the small scenarios of approximately 400 entities. The exercise was coordinated from NRaD.

4.2 DATA COLLECTION METHODS

Three important lessons learned from STOW-E drove the design of the ED-1A data collection system. First, it is essential to have a complete record of all data traffic that passed through the network. This traditional data logger function was necessary to support any type of playback, and to permit investigations into the traffic characteristics in small time intervals. At the same time, however, the project team desired a method for taking real-time "snapshots" of the ACT system's performance. They hoped that these snapshots, when strung together, might provide a coarse overview of the exercise events without requiring the excessive data processing time of the more complete data logger files. The third lesson learned was that both means of data collection must be time synchronized for meaningful comparisons of multiple files. The NTP discussed earlier met this requirement.

The first requirement was fulfilled using the Acusoft data-logging software on an SGI host. To satisfy the second requirement, the project team decided to use SNMP to monitor the ACT system. Management Information Base (MIB) variables were written and incorporated into the HPAG, AT, and AH code to internally monitor performance values of interest. These MIB variables were queried at 1-minute intervals from two network management stations (for redundancy). The project team used management software at these collection points to view the data in near real time while compiling history files that could be analyzed rapidly (as compared to the logger files) to provide a "big-picture" overview of the system's performance soon after the completion of the exercise. Appendix C contains the SNMP MIB for the HPAG, Appendix D contains the MIB for the AT, and Appendix E contains the MIB for the AH.

4.3 ED-1A EXPERIMENTS

As shown in table 1, the basic experimental design for ED-1A can be represented in a three-dimensional matrix design. One dimension was QES; the second dimension was the type of multicasting employed; and the third dimension was the number of participating entities simulated in the test event. QES was on or off; multicasting was either completely off, fully implemented (i.e., an

optimal number of multicast groups used) on the LAN with one group on the WAN, or fully implemented on both the LAN and the WAN. The intersections of these parameters defined a series of experiments. A classical analysis of this data would yield estimates of the main effects of QES and multicasting as well as addressing the existence of any interactions or catalytic effects between the two factors. The design is illustrated in table 1 with references to the size of the scenario employed and the appropriate test event. The two scenario sizes enabled comparisons between the effects of the ACTs on data loads that did not require them, and loads that stressed their performance. Appendix A includes a greater description of the test events and WAN performance information as a result of the events.

Table 1. Test matrix used for the ED1-A experiments.

	No MC	LAN MC/1 WAN Group	LAN/WAN MC
No QES	Large, event 5		Large, event 3
	Small, event 6		Small, event 9
QES	Large, event 4	Large, event 2	Large, event 1
	Small, event 8	Small, event 9.5	Small, event 7

The major problem with using a classical analysis approach was the lack of a truly repeatable scenario. Since ModSAF is a nondeterministic simulation, near-identical starting conditions do not result in identical outcomes or traffic loads. Proper selection of the dependent variable(s) to be compared was another challenge. Each event's results consist of a nonstationary time series of various transmission loads, reception loads, etc. The project team chose arithmetic means and median values, measures of central tendency, high and low values, and standard deviation to describe these time series.

4.4 MULTICASTING

The test strategy included three different multicast configurations to evaluate the behavior of the different configurations.

One configuration was very similar to broadcast in a bridged ethernet. This configuration used only one application-level (LAN) multicast group. All traffic was sent to that multicast group, and as a consequence, every simulator subscribed to this group. Additionally, only one WAN multicast was in use, and all sites were members of it. Thus, every packet on a site LAN was to be forwarded across the WAN to every other HPAG, and every packet received was to be delivered. This configuration essentially "turned off" multicasting, in that there were no efficiency benefits compared to a bridged broadcast case.

Another configuration consisted of "full" use of multicasting. The HPAGs provided could provide up to 65,536 application-level multicast groups for use by simulators and agents. The HPAGs used 127 WAN multicast groups—one for every nonempty subset of the set of seven sites (2^7 - 1). The HPAGs were to forward multicast datagrams to the appropriate WAN group. Data should only be sent where needed and, therefore, delivered to a requesting host at each site where it was received. Simulators were to use on the order of a thousand groups and, thus, not all groups would be joined at all sites.

A third configuration was a middle ground between these two configurations. The HPAGs provided up to 65,536 application-level multicast groups, as in the "full" configuration. They only used one WAN group, however, and each HPAG joined this one group. As a result, the HPAGs were to forward a multicast datagram to this group if any other site needed the data. Thus, data that were purely local would not be forwarded, but other data would be sent across the WAN to all sites. Then, each site would deliver the data to a host if it was needed. With infinite WAN bandwidth and infinite HPAG performance, this scheme, from the simulator's point of view, should be thoretically indistinguishable from the "full" scheme. Furthermore, this third solution replicates the data forwarding mechanism from LAN to WAN that was used for STOW-E with the AG. However, the HPAG can filter WAN-to-LAN traffic, while in STOW-E, the AG could not filter this traffic.

4.5 QES

The project team tested the effectiveness of the QES algorithm by comparing the traffic loads of similar scenarios with the QES algorithm both on and off. Both large and small scenarios were run to investigate the effectiveness of QES as a function of exercise size, and QES was run with multicasting both on and off to check for any interactions.

4.6 SUBSCRIPTION AND FIDELITY

The performance of the Subscription and Fidelity algorithms could not be tested separately from the multicasting function of the ACT system. The fact that entities saw and were seen by other entities within each other's sensor ranges proved that subscription was operating properly. Testing of the Fidelity algorithm was more difficult, but during initial testing, debugging routines were used to observe subscription to both low- and high-fidelity multicast groups, confirming that both groups were used.

4.7 OVERLOAD MANAGEMENT

As previously discussed, due to the absence of the QoS and SRC algorithms, the OM function was reduced to LL, or priority-based dropping of packets, to prevent traffic volume from exceeding an established limit. As the available bandwidth for both the LAN and WAN were sufficient, the OM threshold was set at an extremely high value to prevent its use. As a result, no testing of this algorithm ocurred during ED-1A.

4.8 AT

The AT tests were designed to show the interaction between entities generated on the DIS 3.X LAN with entities generated on the legacy (DIS 2.03) LAN. To confirm proper operation of the AT, testers sought to observe the following indications:

Subscription:

1. Only relevant (area of interest) DIS 3.X entities should appear on legacy simulators.
2. Only relevant legacy entities should appear on DIS 3.X simulators.

Quiescent Entity Determination (QED):

1. Quiescent DIS 3.X entities should appear on legacy simulators.
2. Legacy entity PDUs deemed quiescent should not be transmitted to the DIS 3.X LAN.

3. Active legacy entities should appear on DIS 3.X simulators.
4. Active DIS 3.X entities should appear on legacy simulators.

Expiration Checker:

Expired DIS 3.X entities should be removed from legacy simulators.

Entity Regeneration:

Quiescent DIS 3.X entities should appear on legacy simulators.

PDU Culling:

Local collisions should not be passed across the AT interface.

Due to resource and time constraints, the ED-1A events to test the AT component of the RITN system were not fully implemented. ED-1A event 11.16.95_18.1 was used solely to demonstrate the viability of using the AT to allow legacy (DIS 2.X) applications to effectively participate in a DIS 3.X exercise. To more fully analyze the performance of the AT, a separate test was conducted at NRaD following the conclusion of ED-1A. The configuration included two LANs connected through the AT via separate interfaces. Both LANs supported SGI workstations connected by ethernet, and running ModSAF to generate DIS entities. The following components were used:

1. One ModSAF 3.X (i.e., ModSAF using ACT control packets) graphical user interface (GUI)
2. Eight ModSAF 3.X back-ends
3. One AH
4. One AT
5. One ModSAF 2.X (i.e., legacy ModSAF not using ACT control packets) back-end
6. An HP workstation for real-time data collection of Management Information Base (MIB) variable data

All components were attached to the 3.X LAN except for the single SGI running the 2.X ModSAF, which was connected to the secondary interface on the AT.

The 3.X ModSAFs were run with the following options: turretdr, safritn, qo(quiescent object), envweathernosim, skeptic, and relative_time. The 2.X ModSAF, on the other hand, was run with the these options: noturretdr, nosafritn, noqo, nomulti, nobundle, skeptic, and relative_time.

4.9 TIME SYNCHRONIZATION

The effectiveness of the NTP was confirmed prior to the commencement of ED-1A and was not tested directly during the demonstration.

4.10 QUALITY OF SERVICE

The QoS was written by Bolt, Beranek and Newman (BBN), but was not implemented for ED-1A since it was not supported by the switch manufacturers. As a result, it was not tested in ED-1A.

4.11 NETWORK IMPLEMENTATION

It was anticipated that difficulties might be encountered as the hosts and routers were required to deal with high packet volumes on large numbers of simultaneously active multicast groups. The project team conducted tests to find the "break points" of the various implementations of multicast in the routers and hosts with different numbers of multicast groups. Once break points were found, the router implementations were further characterized for how well they continued to perform when pushed beyond the bounds of flawless operation. The results of the tests will determine the operational area of the routers in the ED1A test bed.

4.11.1 Router Multicast Forwarding Test Bed

The initial router multicast forwarding performance testing was performed using the following test bed configuration. Subject routers were configured for multicast routing between two separate ethernet segments. Hosts were configured to serve as multicast IP traffic sources and sinks using NRL's Multicast Generator (MGEN) software test tools. Figure 11 depicts the test setup.

The Sun SPARC 20 hosts generated traffic on one side of the router while the SGI machines joined multicast groups and log traffic on the other. The network's general ethernet sniffer monitored traffic on the ethernet segments to ensure that packets lost to excessive collisions or CRC errors did not distort observations.

4.11.2 Host Fiber Distributed Data Interface Performance Evaluation

In preparations for the RITN ED-1A exercise, there were some problems with multicast performance on hosts using Fiber Distributed Data Interface (FDDI) interface cards. Additional tests using the MGEN tool set evaluated the host's capability to generate and receive multicast traffic while attached to an FDDI LAN, as shown in figure 12.

4.11.3 Router Test Procedures

MGEN scripts were created for unicast and multicast traffic generation. Multicast traffic was generated in successive tests on different numbers of simultaneously active multicast groups. When multiple groups were involved, the traffic load was distributed evenly across the groups. The routers' ability to forward the aggregate traffic load without packet loss was evaluated.

The ethernet sniffer and MGEN logging tools were used to perform these tests under different loading situations with varying numbers of multicast groups (up to 1000 simultaneously active groups). Multiple SGI workstations running the MGEN logger (DREC) were used to log the aggregate packet load by each workstation recording traffic on different multicast groups. The network's general sniffer was used to perform packet counts when the aggregate rate was higher than what the available workstations could log. In this case, the workstations running DREC were still used to ensure that no groups were missing (i.e., router was not failing to forward packets for an active group) in the received traffic.

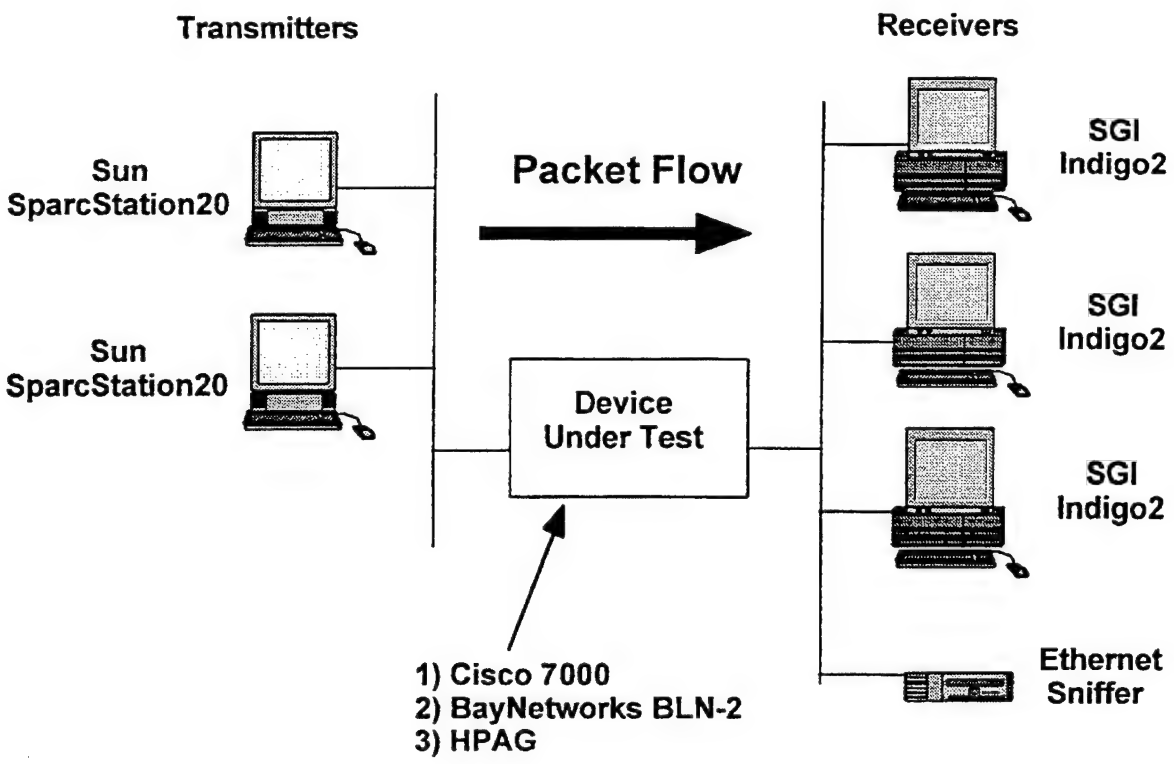


Figure 11. Multicast performance test bed.

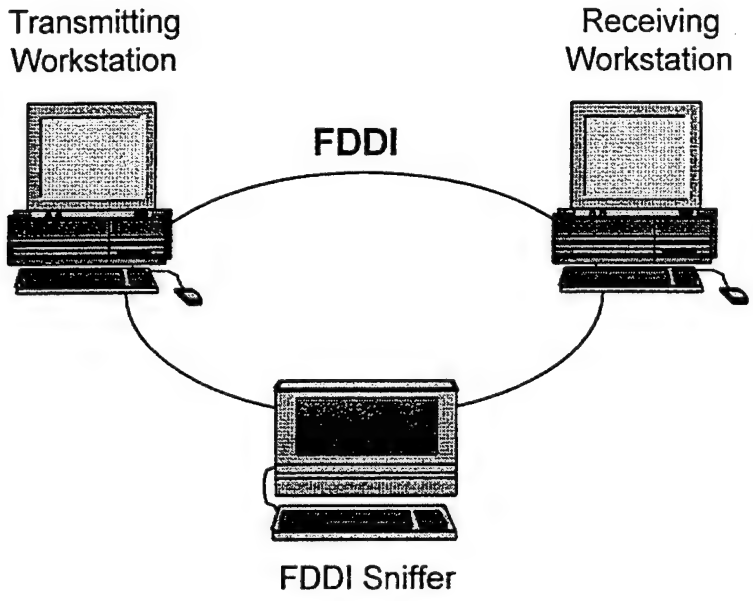


Figure 12. FDDI testing setup.

5. ANALYSIS AND EXPERIMENTATION

Experimentation of the various aspects of the RITN system was conducted before, during, and after the ED-1A demonstration. What follows is a description of this testing and direct analysis of the results observed or recorded.

5.1 WAN MULTICASTING

5.1.1 Overview of Multicast Delivery Data from LAN-WAN-LAN

The data from the HPAGs were examined to determine the amount of data sent from each site to each other site as a function of time. Various ratios were then calculated, such as the fraction of a site's outgoing data sent to each other site. Such information is interesting because it shows the communications patterns among sites and the effect of multicasting on reducing the delivery of data among the sites.

Each HPAG maintains a counter for each WAN group of packets sent to that group. These counters were sampled once a minute via SNMP. Thus, the number of packets sent to each WAN group in each minute can be calculated. For a given HPAG, the total number of packets sent to the WAN can be computed. The per-WAN group table was also examined to compute the number of those packets destined for a particular site, since it was known which WAN groups each HPAG had joined. Doing this for each site yielded the number of outbound packets destined for each site. The sum of these counts will generally be greater than the total number of packets sent, since many of the packets will be delivered to more than one site. Then, the number of packets sent to each site can be divided by the total number of packets, to determine the fraction of WAN-bound traffic sent to that site.

A large number of graphs in Appendix B were produced of values derived from the SNMP MIB data. These range from simple calculation of data rates to complex derived values, such as the fraction of outgoing traffic sent to a particular peer. While examining these graphs, it is important to keep in mind that the MIB variables were collected at nominal 1-minute intervals. Thus, all of the derived rates are nominally 1-minute averages. Because they are 1-minute averages, and not peak rates over some small interval related to the maximum allowable latency, they cannot be directly used to size communication bandwidth.

Graphs were produced of 1-minute averages of LAN input packet rate, LAN output packet rate, WAN input packet rate, and WAN output packet rate. Graphs were also produced for the WAN-bound forwarding ratio, which is the WAN output rate divided by the LAN input rate. This number is the fraction of multicast packets that originated on the LAN and transmitted to the WAN. Table 2 lists the general patterns of the graphs in Appendix B.

Table 2. General patterns from Appendix B graphs.

Number of Multicast Groups*	Ratio of Packets Transmitted to the WAN to Packets Generated at a Site	Ratio of Packets Received by a Site to Packets Delivered to the LAN
Optimum no. on LAN/127 on WAN	0.8:1	1:1
Optimum no. on LAN/1 on WAN	Almost 1:1	0.9:0.2
1 on LAN/1 on WAN	Almost 1:1	1:1

*The optimum number of multicast groups on a LAN varied by site and by conditions.

In most cases, substantially all of the LAN data were forwarded to the WAN. In a few cases, with many LAN groups and 127 WAN groups, an HPAG forwarded most, but not all of the local multicast traffic. An example is IDA during event 9.1 on November 15, which forwarded 80% of the local multicast traffic to the WAN. This result indicates that most simulation traffic was requested by at least one other site. It must be remembered that applications like PVDs and WAVs request traffic for the whole large area of view and subscribe to every site's set of multicast addresses.

With many LAN groups and 127 WAN groups, substantially all of the data received from the WAN by a site HPAG was delivered to the site LAN. This is as expected because the bi-level protocol communicates which groups are needed at a site to all other sites. The WAN directed the traffic so that only the sites that need the data received it. Since 127 was an adequate number of WAN groups to distinctly address all subsets of the seven sites, only the data which were subscribed to by the site were forwarded to the site through the WAN.

In addition to MIB variables counting inbound and outbound packets per interface, MIB variables were monitored that counted packets sent to the WAN on a per-WAN-group basis. From these values, the packet rate from each HPAG to each other HPAG was computed. That is, a packet was counted as being sent from A to B if A sent it to a WAN group to which B subscribed. The fraction of data sent to a particular site compared to the total data from that sender is computed additionally, and is shown in Appendix B.

With many LAN groups but only one WAN group, a variable fraction of the data received from the WAN was delivered to the LAN. This, too, is expected because almost all data were forwarded to the WAN, but much of the data were not needed at each site. Thus, the data were dropped upon receipt from the WAN by the HPAG to the LAN. From this test, the project team concluded that multicasting reduced traffic to the sites by a factor of 3 and 5 times when compared to a pure broadcast scheme (one LAN group and one WAN group). It is noted that the amount of reduction is skewed in table 2 because the NRaD site was the only one to host the Blue Forces. Everyone else on the network interacted with NRaD because they were supporting various components of the Red Forces.

With one LAN group and one WAN group, all the data were delivered like the DIS 2.X broadcast paradigm. This is as expected because all simulators subscribed to the single LAN group in use.

5.1.2 Communication Pattern Among Sites Related to the Scenario

The project team computed the fraction of data sent to each site compared to the total data sent to the WAN by all other sites. This number represents the fraction of WAN data actually sent to that site compared to how much data the site would have received under a broadcast scheme. These graphs are shown in Appendix B. The graphs of data rates between sites that are derived from the per-WAN-group statistics are very useful for insight into the effect of scenario choice upon communication patterns.

To illustrate the results of multicasting, figure 13 shows several graphs that describe the WAN performance between the network sites during Test Event 1. This test used both WAN and LAN multicast with all ACT algorithms for a large scenario. The upper left graph shows the LAN input rates for the different sites, which can be considered the forcing function for the network. The upper right graph shows the WAN-bound forwarding ratio, or the number of packets sent to the WAN divided by the number created on the LAN. The value can be greater than 1 due to time differences in sampling the different counters. The middle graph shows the LAN-bound forwarding ratio, which is the

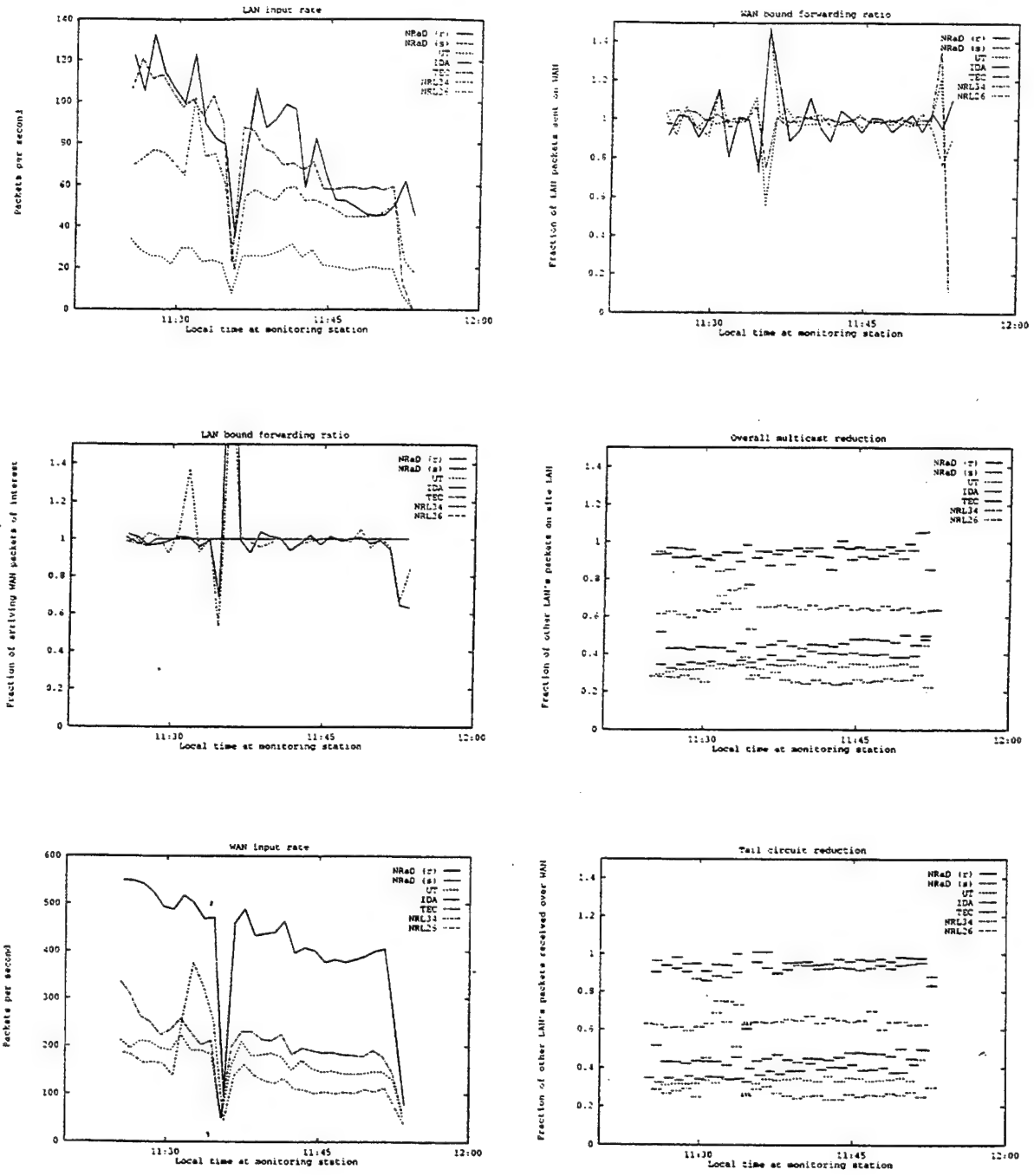


Figure 13. Charts of network performance during Test Event 1, using the large scenario, all ACT algorithms, and WAN and LAN multicast.

number of packets sent to the LAN divided by the number received from the WAN. The lower left graph shows the WAN input rate and the lower right shows the tail circuit reduction rate, which is the amount of traffic delivered to the site divided by the traffic that would have been delivered to the site in a broadcast communication scheme. From the graphs of figure 13, LAN input rates ranged from 30 to about 100 packets per second (pps), with peaks at 130 pps. WAN input rates were about 500 pps for one NRaD site and about 200 pps for most of the other sites. The LAN-bound forwarding ratio was close to one that indicates the data were sent to the correct sites, and that the multicast routing protocol functioned correctly. Overall and tail circuit reduction rates were comparable except for the NRaD sites. They received 90% of the total traffic load while the other sites received 20% to 70% of the total traffic load. Finally, the dropout at 1100 has no explanation, but it appears not to be a function of the multicasting or HPAG performance.

The most important aspect of the scenario as to the WAN multicast performance is that one must be careful how the simulation forces are distributed within the simulation scenario, and how they are distributed for processing in the physical network. The testing here proves the team's assumption that the WAN-loading benefit of multicasting can be defeated by distributing the forces such that all WAN site nodes need to have an interest in all the other nodes. This is the case with NRaD, as shown by the previous example where NRaD received 90% of the broadcast traffic. NRaD supported all of the simulated entities on the Blue Force. All Red Force engagements required data from NRaD.

5.1.2.1 Small Scenario. Patterns in data rates in the small scenario tests were far less evident, to the point where it is difficult to impossible to identify them. The scenario called for activity to take place in a limited geographic area, and test 7.1 on 16 November shows that TEC, NRL34, and NRL26 received less data than the other sites. Because of the limited geographic area, it was not surprising that there was no vast separation of traffic by sites.

5.1.2.2 Large Scenario. The graphs from the large scenarios, Test Events 1 through 5, show that many sites sent most of their data to one or more of the NRaD sites, which jointly contained all Blue Forces. Many sites sent fairly small fractions of their data to some other sites, and the patterns sometimes changed during exercises.

Examining these graphs together with a map of the virtual location of each site's forces, it is evident that sites with forces far apart sent very little traffic to each other. Sites with Red Forces sent most of their traffic to the NRaD site containing the adjacent Blue Force.

For example, in test MAX on 16 November, "box10," the HPAG at NRL34, sent the largest fraction of its data to "suns," the NRaD HPAG carrying the traffic for the adjacent Blue Force. The next largest fractions of data were sent to "lonestar" (NRL26), the HPAG carrying the network traffic for the adjacent Red Force, and "rockets," the NRaD HPAG servicing traffic for the other Blue Forces. Small amounts of data were sent to TEC, IDA, or UT, although more data were sent to IDA towards the end of the exercise.

During this same exercise, "minnie" (an IDA HPAG) sent large amounts of data to both NRaD sites, and fairly little to the other sites. This corresponds very well with the scenario that called for Red Forces at IDA to engage Blue forces at NRaD. It is obvious from the results that the system was forwarding the information to the sites only where the information was needed.

Although it is difficult to state an overall, exercise-wide "reduction factor" due to multicasting, it is clear that multicasting was very helpful at each site. Though the use of multicasting resulted in little data reduction to the sites hosting the Blue Forces in the large scenario, this was not a failure of multicasting; these sites subscribed to this data because they were heavily involved. In the MAX test, sites other than NRaD generally received 35% to 60% of the traffic they would have received without multicast. In test 10.1 on 16 November, sites generally received 20% to 50% of the data they would have received without multicasting.

In addition to these significant reductions of data, it is important to observe that some sites received a small fraction of the data during entire runs. In the 10.1 test on 16 November, the HPAG "raphael" (TEC) received 25% of the total traffic sent by all other sites. This implies that with careful attention to scenario design, it should be possible to have sites with dissimilar bandwidth connections participate in simulations.

5.1.3 Bi-level Multicast Implementation

The team examined traces of bi-level control traffic captured from the ethernet between one of the NRaD HPAGs and the site router. In addition to the periodic messages sent by the protocol (acknowledgment and checksum of current state), the team observed a number of "join/leave" messages from various HPAGs informing the others of changes in the list of locally needed groups. Other than the periodic messages and the join/leave messages, the team observed very few other messages. This indicates that the protocol worked as intended, and rarely invoked recovery mechanisms.

In one exercise, 11.16.95_3.1, the team saw many messages relating to the recovery mechanism. Closer inspection of traces from this run revealed that NRaD's "A" LAN continued to send acknowledgments that it stopped receiving packets from other sites at approximately 10:08:37 UTC. The NRaD-A LAN continued to send data messages, and other sites continued to send acknowledgments reflecting receipt of those messages. All sites other than NRaD-A began to retransmit data. Observation of simulation behavior via the teleconference confirmed that remote simulators were no longer appearing at this site, but that NRaD-A's simulators were still appearing elsewhere. While a particular network problem (a transmit-only site) caused significant problems, it is interesting to note that control traffic among the other six sites still proceeded normally, and no anomalous simulator behavior was observed.

The HPAGs periodically send checksums of their current states; these are checked by other HPAGs to verify the consistency of the state they hold regarding their peers. If these checksums do not match, a message requesting a complete transfer of state is sent. In all of the experiments examined, except for 6_1 on 16 November 1995, the team observed almost no messages indicating checksum mismatches. During this run, evidence was found indicating that NRaD-A stopped receiving packets on many occasions. During this run, the HPAG at NRaD-A was restarted on several occasions because it was conjectured that the router was not receiving join messages for the WAN groups. On several occasions, the problem returned precisely 4 minutes after a restart, which is the time it takes the router to deliver a group join message after receiving a report. It is believed, therefore, that the operating system on the HPAG sent group membership reports when the HPAG program started, but did not refresh these messages as it should have. This problem was not observed at other sites.

Despite attempts to design the bi-level protocol and implementation to avoid self-synchronization, the team observed a small degree of self-synchronizing behavior. This was due to an implementation choice rather than being an inherent property of the bi-level protocol design.

During normal operation, the HPAGs send time-stamped control messages. On receipt of these messages, they compute the one-way delay between the sender and the receiver. These computed delays are included in the acknowledgment control messages. Due to the message format, these computed delays have a resolution of roughly 4 msec. The team examined graphs of these one-way delays between HPAGs, and observed periodic spikes in the delay distributions. The base delays were approximately half the round-trip delays observed via "ping" on the test bed; this is as expected. Many of the reported large delays were on the order of 100 msec, and some were as high as 200 msec. The base one-way delays varied by sites, but were typically 40 msec from San Diego to Washington or UT, roughly 30 msec from UT:ARL to Washington, and on the order of 4 to 8 msec within the DC area. As a result of the HPAG's time delay, measurements from control traffic is that of the backbone routers and the ATM switches. The HPAG introduces very little delay. The jitter or variation in delays between HPAGs was very small and on the order of the resolution of the system of 4 msec from coast to coast. The ATM service contributed very little jitter and very little additional delay.

Closer examination revealed that the high-delay messages were associated with transmission of checksums by other HPAGs and, further, that all HPAGs tended to send checksums at roughly the same time. The protocol calls for randomized delays between checksum transmission as well as between transmissions of acknowledgments. While the inter-acknowledgment delay was uniformly chosen from an interval between 1 and 1.25 times the nominal interval (250 msec), the checksums were transmitted with every 100th acknowledgment. The randomization present in the sum of 100 uniformly distributed times was not sufficient to overcome the self-synchronizing effect of the receipt of checksum messages. This is not considered a serious effect, and will be easily corrected in subsequent implementations of the bi-level multicast protocol.

The team examined the bi-level multicast control traffic and extracted application-level multicast group join and leave information. Graphs showing time on the horizontal axis and group number on the vertical axis were drawn, and several trends were clearly visible. The simulators were, indeed, making use of multicasting.

At the beginning of each test run, a large number of joins were reported, as expected. At other times, a smaller number of joins occurred. These are often followed by a similar number of leaves within a few minutes. This is the pattern expected by simulators with entities that move across the virtual geography, given the grid scheme in use to assign multicast addresses.

Meaningful quantitative results cannot be drawn from this data of group joins and leaves because it is very scenario and grid-segment specific. However, it is believed that the results clearly show that the simulators were using the multicast service consistently with the employed grid scheme. That is, as new groups are joined, other groups are dropped since the old groups are out of the region of interest.

5.2 APPLICATION MULTICASTING

The ED-1A tests focused on measuring the performance enhancement due to multicast-based relevance filtering. This portion of the multicasting tests focused on the use of multicast addresses for application-level communication. The application multicast algorithm functions independently of the bi-level algorithm for the employed WAN site-to-site multicasting. The WAN multicast is transparent to application implementation. However, the WAN multicast is dependent on the application-level multicast to determine the network site-to-site interest in data. Several key issues were evaluated for this analysis. The main issue was to determine how well multicast relevance filtering reduced the processing of irrelevant data at the application level because the application-level multicast will reduce the network traffic's impact on the kernel processing. A related issue was to examine the amount of excess multicast packets that were rejected in the kernel, and to comment on the utility of hardware filtering of multicast packets. Finally, the number of multicast groups per application was examined to determine the utilization of this relatively scarce resource of multicast addresses. In all cases, the team was to evaluate the relevance filtering algorithm's performance as a function of the grid sizes of the low- and high-fidelity multicast groups that corresponded to the low- and high-update rate multicast groups.

5.2.1 Application-Level Multicast Effectiveness

Application-level multicast effectiveness was measured by comparing the delivered multicast Entity State PDU (ESPDU) flow to an approximation of the potential broadcast ESPDU flow, had multicast relevance filtering not been implemented. This calculation results in a measure of flow reduction achieved by multicast relevance filtering. Another measure of effectiveness was calculated by comparing the delivered multicast ESPDU flow with that of an ideal filter that delivers ESPDUs to applications with entities that require the ESPDU since it occurs within their region of interest. This calculation gauges how close the relevance filtering algorithms approach the limit of traffic that must be delivered. It shows how much benefit can be gained through better relevance filtering techniques.

Performance of each of the selected relevance filtering tests was assessed at the application level by calculating the ESPDU traffic flow recorded during the experiment and comparing it with the ESPDU traffic delivered to, or flowing on the LAN. Traffic was delivered by the WAN if a multicast subscription existed for at least one application on the LAN. This approximates a comparison of the delivered multicast flow with that of the potentially delivered broadcast flow to determine a measure of the effectiveness of the multicast relevance filtering algorithm. While this approach overlooks the contribution of CP traffic to simulation overhead and the benefit obtained by multicasting Transmitter and Signal PDUs, it still provides a good measure of the effectiveness of grid size on reducing the flow of irrelevant data to each application.

Performance of each of the selected relevance filtering tests was also assessed at the application level by calculating the ESPDU traffic flow recorded during the experiment and comparing it with the calculated traffic flow for an "ideal" relevance filter. The ideal relevance filter calculates the absolute minimum ESPDU traffic that must flow to maintain a consistent state within the simulation. This is determined by accounting for the delivery of each ESPDU from an entity somewhere on the WAN to every host with at least one entity that has that remote entity within its radius of interest (ROI). This amounts to essentially an r-squared test for potential visibility to determine if the PDU must be received. Note that this filter is not practical to implement; it is used as a basis for normalization of the different events. While this approach also overlooks the contribution of CP traffic to

simulation overhead and the benefit obtained by multicasting Transmitter and Signal PDUs, it provides a measure of how well multicast relevance filtering reduces the flow of irrelevant data to each application. Normalization of the application-level ESPDU flow for each multicast test event against its calculated ideal filtered traffic flow facilitates direct comparison of the different experimental tests.

Figures 14 through 17 plot and compare filtered ESPDU flow for typical applications on the NRaD-A LAN; one plot is shown for four multicast grid-size test events. Each plot illustrates a time-history for a representative application: the LAN "broadcast" flow, the ESPDU flow delivered due to the applications' multicast subscriptions, and the calculated ideal filtering ESPDU flow. The four hosts were chosen to be representative of the behavior typical of applications during the test event. Due to different scenarios and distributions of entities, the host traffic is not directly comparable.

To summarize the application-level performance of multicasting with the grid-size filter, the team used the following methodology for each test event. The log files were processed to calculate the broadcast ESPDU flow, the multicast ESPDU flow, and the ideal filtered ESPDU flow for 10-sec intervals. The ideal filtered flow is based on geographical distance between entities within each entities region of interest. The ideal filter forwards information only to the entities that need to see the information based on a circular region of interest and no more. This represents the best or ideal data-forwarding solution. The real-time implementation of the algorithm has not been applied, but the performance of the ideal filter is easily postprocessed from data logger state information using the geographical distances between the entities. For each interval, the ESPDU flow reduction due to multicasting relative to broadcast was calculated as

Flow Reduction = (LAN Broadcast ESPDU Flow Grid Filter ESPDU Flow)/LAN Broadcast ESPDU Flow

Also, for each interval, a measure of "excess flow" was determined by normalizing the multicast ESPDU flow that exceeded the ideal filter flow, as

Excess over Ideal = (Grid Filter ESPDU Flow Ideal Filter ESPDU Flow)/Ideal Filter ESPDU Flow

This calculation measures the "ineffectiveness" or potential improvement that the relevance filtering algorithms have yet to achieve. Thus, for each time interval, a measure of multicasting effectiveness was calculated with reference to the broadcast flow, and a measure of the multicasting "ineffectiveness" was calculated relative to the ideal filtering case.

Next, a probability distribution of ESPDU flow reduction per unit time was calculated, as shown in figure 18 for three different hosts. To understand this graph, consider the right-most line representing the multicasting performance of a host in the exercise. The line breaks away from the 100% line at the top for the percentage of time the reduction level was noticed with approximately a 35% traffic reduction. This means that from the samples taken, this host transmitted less often as a result of multicasting 100% or all the time. As one follows down the right-most line, 75% of the time, multicasting reduced traffic from the host approximately 65%. Continuing down the right-most line, 25% of the time, the traffic reduction from this host was reduced about 77%. Finally, about 2% of the time, traffic was reduced 83%. The vertical lines that intersect the 50% line represent the median traffic reduction per time interval of 10 sec. The shaded regions represent the spread about the median (25% to 75% occurrences) for these three single hosts in one test event. These 25% to 75% statistical spreads about the median may be calculated for each host processor in a test

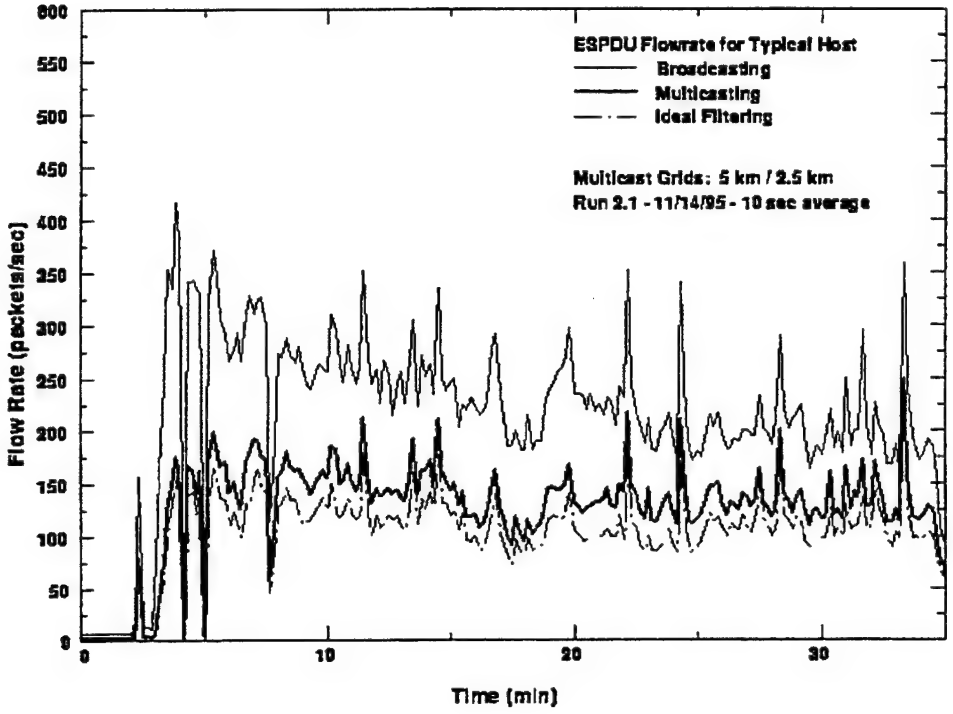


Figure 14. Application-level ESPDU flow multicasting Test Event 2. Note the thick curve is ESPDU flow multicast to 5-km low-rate and 2.5-km high-rate grid.

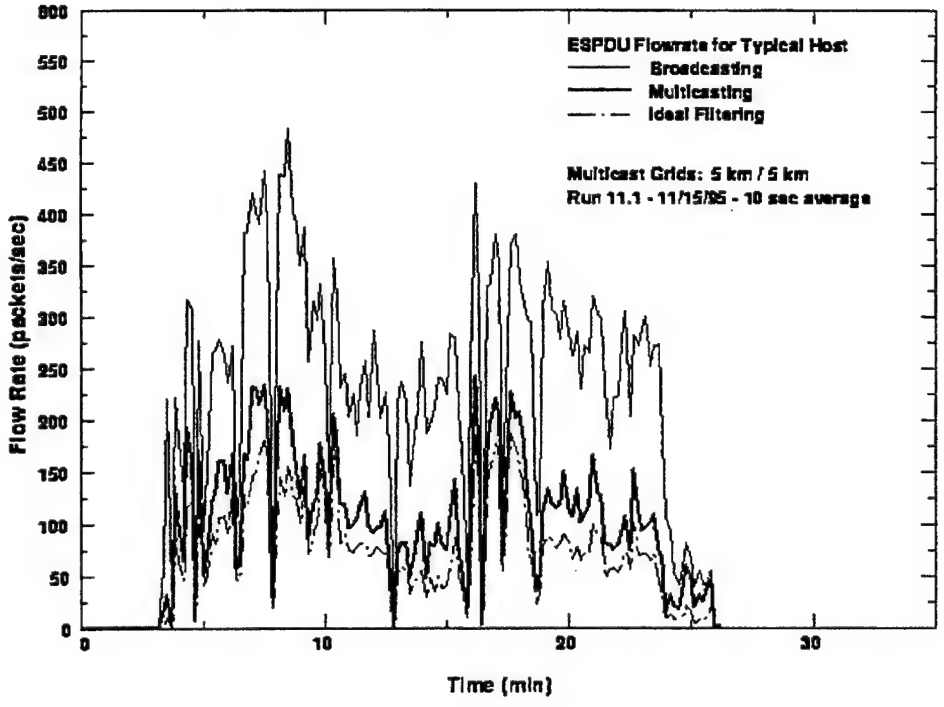


Figure 15. Application-level ESPDU flow multicasting Test Event 11. Note the thick curve is ESPDU flow multicast to 5-km low-rate and 5-km high-rate grid.

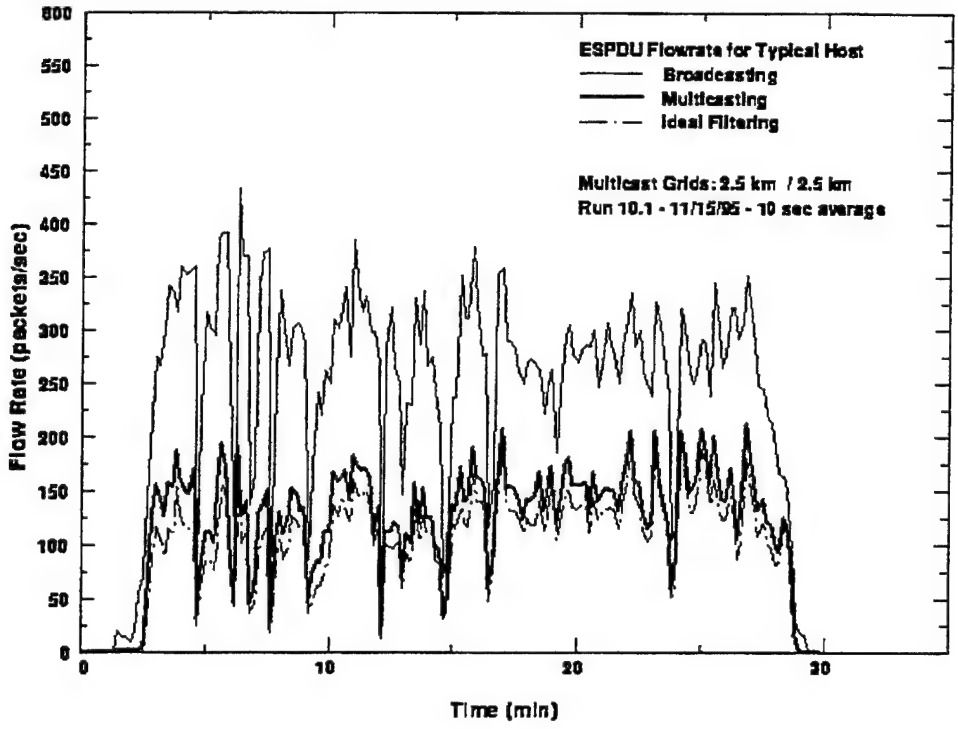


Figure 16. Application-level ESPDU flow multicasting Test Event 10a. Note the thick curve is ESPDU flow multicast to 2.5-km low-rate and 2.5-km high-rate grid.

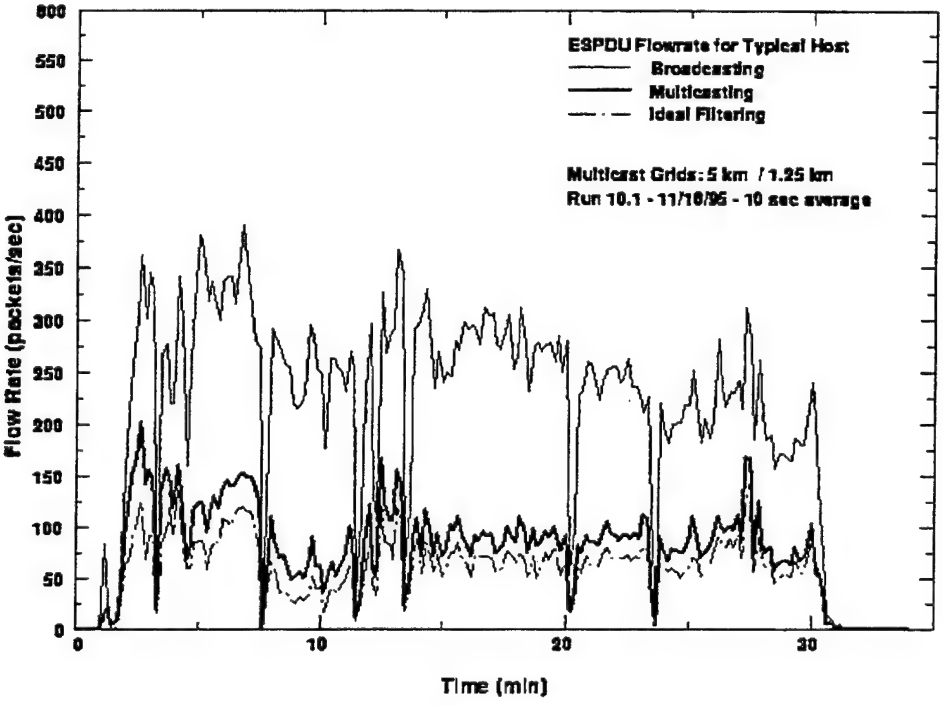


Figure 17. Application-level ESPDU flow multicasting Test Event 10b. Note the thick curve is ESPDU flow multicast to 5-km low-rate and 1.25-km high-rate grid.

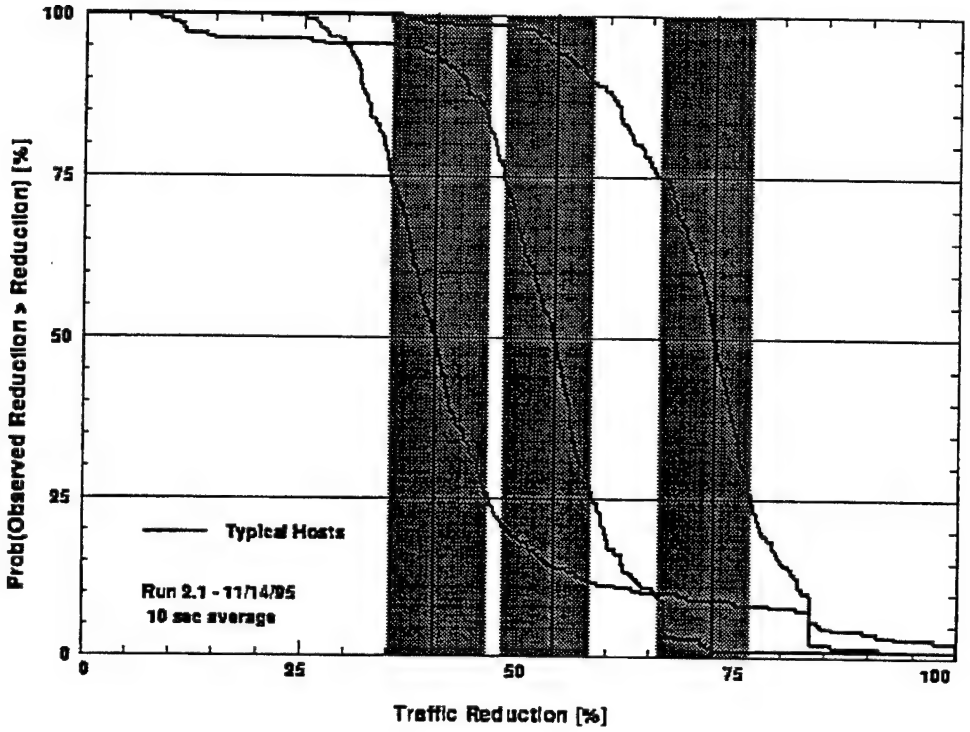


Figure 18. Probability distribution of flow reductions per time interval for typical hosts; broadcast ESPDU flow compared to multicast ESPCU flow. The three lines represent 25% occurrence, median occurrence, and 75% occurrence.

event, as shown in figure 19. The top of the shaded line in figure 19 corresponds to the level of traffic reduction from multicast observed 25% of the time, and the bottom representing the traffic reduction 75% of the time, with a median at the solid line. Finally, these summary statistics are calculated by averaging the reported median flow rates for each application to acquire a single site-wide or population flow reduction statistic. Then, the team calculated the median ESPDU flow reduction for each test event by summing up the site levels, as listed in table 3. This calculation was repeated for each combination of high- and low-resolution multicast grid sizes.

Table 3. Percent ESPDU flow reduction at the application level.

Application-Level % ESPDU Flow Reduction			
Filename/Date	**Low-/High-Rate Grid Size**	**Mean/StDev**	**Median**
11.14.95_2.1	5 km/2.5 km	52.5/13.0	49.2
11.14.95_1.5	5 km/2.5 km	47.7/15.7	49.8
11.15.95_11.1	5 km/5 km	55.3/7.4	56.2
11.15.95_10.1a	2.5 km/2.5 km	50.0/10.3	51.3
11.16.95_10.1b	5 km/1.25 km	54.8/19.9	59.8

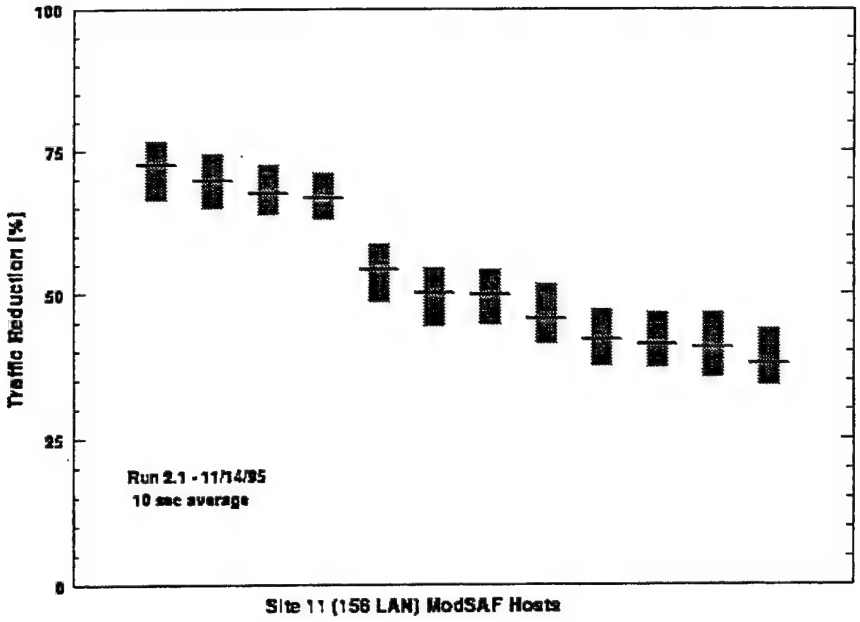


Figure 19. Bar plot summarizing application-level ESPDU flow reduction for multicast vs. broadcast case where the shaded regions represent the 25% to 75% distribution about the median for several ModSAF hosts.

The same procedure may now be applied to the excess flow statistics to obtain the relative packet flow with respect to that required by ideal filtering. This is shown in table 4. Using this method, it is now quite reasonable to compare dissimilar scenarios with differing numbers of entities. Normalizing the traffic flow at each application by its own ideal filtered flow essentially takes into account differences due to the scenarios and entity count.

Table 4. Percent excess ESPDU flow at the application level.

Application-Level % Excess ESPDU Flow			
Filename/Date	**Low-/High-Rate Grid Size**	**Mean/StDev**	**Median**
11.14.95_2.1	5 km/2.5 km	26.7/9.6	22.9
11.14.95_1.5	5 km/2.5 km	39.5/18.1	38.2
11.15.95_11.1	5 km/5 km	46.1/20.5	44.4
11.15.95_10.1a	2.5 km/2.5 km	29.1/11.1	29.5
11.16.95_10.1b	5 km/1.25 km	29.6/12.6	33.3

Note that this analysis methodology is an attempt to quantify performance on an application level. The summary statistics calculated above and listed in the tables are all population statistics. That is, the temporal statistics reflecting flow reduction or excess flow per application were combined with the performance of other applications to obtain the mean performance of the population. This allows generalization of performance trends, providing a broader basis for comparison.

Reviewing the results in tables 3 and 4, the following observations can be made about the effectiveness of various multicast grid sizes for the different test events. Increasing grid sizes above the calculated optimum (5 km for high-rate grid and 2.5 km for low-rate grid in ED-1A) generates increased levels of excess ESPDU traffic, as compared with the ideal filtering model. In Test 11, the high-rate grid was increased from 2.5 km to 5 km, and a larger ESPDU flow was observed. A larger high-rate grid implies that as an entity's ROI intersects a new grid square, traffic from the entire grid square will be routed to that entity. This fits with the geometry of the situation and conforms to expectations.

The benefits of decreasing grid sizes, in either Test 10.1a with the 2.5-km low-rate grid or in 10.1b with the 1.25-km high-rate grid appear to be somewhat ambiguous. In theory, reducing the grid size would more closely approximate the ROI of each entity and the collective ROI of each application. Actually, the smaller low-rate grid in Test 10.1a should have little measurable impact on performance due to the low flow rate of data to those groups. That the small high-rate groups in Test 10.1b exhibited no measurable improvement in performance is surprising, since fundamentally, smaller grids will deliver less irrelevant data to applications.

There are several possible explanations for this apparent inconsistency. First, note the wide disparity in results of the two tests that had 5-km/2.5-km grid sizes. Their application-level excess ESPDU flow has a 10% difference, although it is within the standard deviation of the tests. This implies that the unmeasurable difference between the two small grid tests may really be insignificant, or things may be much more than they appear. A second point is related to a limitation of this analysis. The measure of excess ESPDU flow does not consider the overhead required to manage the different-sized multicast grids. Smaller grids will require more overhead via multicast group join packets and related messages. Additionally, although QES was held constant (QES On) for these runs, smaller grids, especially high-rate grids, should require more overhead. In short, this analysis of excess ESPDU flow does not tell the whole story; a more detailed review of all traffic types needs to be completed.

If overhead does increase with smaller grid segments, it is hypothesized that circular ROIs should be used for subscription instead of rectangular ROIs for sensors that can be modeled with a circular ROI. This should result in more effective filtering. Though not confirmed in ED-1A, circular ROIs will allow fewer groups to join because the rectangular ROI corners will not be joined, reducing overhead. The larger the region of interest, the more pronounced the benefit should be, especially with the small-terrain grid segments. Future investigation should include the use of the circular ROIs.

Finally, the multiple-grid format of the multicast algorithm complicates this analysis. While the grids were co-registered for high and low fidelity, different-size grid segments lead to unexpected edge-effect issues when measuring filtering performance that uses the different proportional regions of interest between the two grid formats. Also, this analysis is based on a population statistic to facilitate cross-event comparison. Since the simulation application results are nonrepeatable, there is a wide range of variation in entity distribution among the grid segments for each test event. Therefore, it is difficult to make definitive conclusions about application network traffic reduction due to multicasting.

5.2.2 The Case for Hardware Multicast Filtering

Current workstations used for simulation (e.g., SGIs) are limited in their ability to filter out unwanted multicast packets in their network interface hardware. Most interface hardware supports schemes for hashing multicast addresses to a relatively small number of bins and accepting/rejecting packets based on bin number. Some workstation vendors do not even take advantage of this crude level of filtering in their network drivers. For those that do, the large number of groups and relatively complex mappings employed for relevance filtering make this sort of approach ineffective.

Consequently, most multicast packets on an ethernet LAN are actually received and processed by a workstation's kernel. Those packets destined for groups that the workstation is a member of are passed to the subscribing application. Those destined for other groups are rejected, but only after an interrupt has been generated, and the kernel driver/protocol code has run. This is a waste of workstation processing power that could be eliminated if workstation vendors' products incorporated network interfaces capable of hardware filtering. The benefit of this type of hardware filtering is that the unwanted multicast packets would be rejected without interrupting the kernel.

In an attempt to quantify the benefits of hardware filtering, the multicast test events were analyzed in a manner similar to the calculation of flow reduction described above. The files were processed to calculate the flow of unique ESPDUs to the site and the multicast ESPDU flow to each application for 10-sec intervals during the analysis windows stated in the table earlier. The unique ESPDU flow is calculated by determining that at least one application subscribed to the multicast group for each ESPDU; this is equivalent to the LAN broadcast ESPDU flow. For each interval, a "hardware-filtered" value was determined by normalizing the amount of traffic that exceeded the multicast flow by the multicast flow as

(LAN Broadcast ESPDU Flow Grid Filter ESPDU Flow)/Grid Filter ESPDU Flow.

The statistics for application-level hardware filtering in table 5 appear to be fairly well distributed. This is due in part to some applications sharing overlapping regions of interest, while others have little commonality with the applications at the site. Thus, several of the applications that are receiving the same flow will appear to have little measurable hardware excess, while applications with little commonality will have a tremendous performance difference. Thus, when measuring the amount of rejected multicast packets, it is obvious that some applications are spending a very significant fraction of their time rejecting unwanted multicast traffic. The benefits of hardware filtering of multicast packets are obvious. More capable hardware filtering of multicast packets could reduce the number of packets handled by many hosts by a factor of 2 or more, judging by the data in table 5.

Table 5. Percent hardware-filtered ESPDU flow at the application level.

Application-Level % Hardware-Filtered ESPDU Flow			
Filename/Date	**Low-/High-Rate Grid Size**	**Mean/StDev**	**Median**
11.14.95_2.1	5 km/2.5 km	128.4/72.9	96.8
11.14.95_1.5	5 km/2.5 km	108.2/63.8	100.4
11.15.95_11.1	5 km/5 km	129.5/39.0	128.5
11.15.95_10.1a	2.5 km/2.5 km	108.0/41.7	105.8
11.16.95_10.1b	5 km/1.25 km	172.3/144.6	145.6

5.2.3 Group Usage

The packet flow rate is not the only important performance issue to observe from these test events. Earlier it was mentioned that smaller grids might cause higher overhead either when sorting received multicast packets or when joining new groups as entities move. The calculation of groups in use per application in table 6 confirms intuition that smaller grids will require applications to join larger numbers of multicast groups. These numbers are dependent on the average numbers of entities per application and their scenario and distribution in the battlespace. The numbers of multicast groups joined per application are well within the capabilities of the current workstations. This chart of increasing group usage with decreasing grid size reflects one of the factors inherent in the use of smaller grid sizes; the required number of groups will eventually exceed the capabilities of the workstations. Conversely, larger grids will require smaller numbers of multicast groups, as expected. Therefore, a balance must be reached, and an optimum grid spacing determined through experimentation or modeling before commencing of a major exercise. The optimum grid sizing approaches relevance filtering of the ideal filter while minimizing the impact of the number of grids on subscription overhead.

Table 6. Per-application multicast group usage.

Per-Application % Multicast Group			
Filename/Date	**Low-/High-Rate Grid Size**	**Mean/StDev**	**Median**
11.14.95_2.1	5 km/2.5 km	151/33	151
11.14.95_1.5	5 km/2.5 km	144/34	135
11.15.95_11.1	5 km/5 km	66/16	72
11.15.95_10.1a	2.5 km/2.5 km	226/55	227
11.16.95_10.1b	5 km/1.25 km	414/163	411

5.3 QES, SUBSCRIPTION, FIDELITY, AND MULTICASTING ANALYSIS AND INTERACTIONS

Due to the interrelationships between QES, subscription, fidelity, and multicasting, this section attempts to analyze their performance and interactions as a whole.

Also, it must be noted here that CP implementation for QES was robust and did not exhibit instabilities or breakdown. During events of transient high loss rates, episodes of network connectivity loss, and heavily overloaded simulations, the CP maintained itself.

5.3.1 Selection of Runs for Analysis

The results described in this section are based on analysis of PDUs recorded by a data logger located on the NRaD-A LAN. While these log files do not provide complete information on all entities, the presence of a PVD that was zoomed-out to view the entire battle ensured that at least low-fidelity data were available on all entities that were situated on the playing field; this information is sufficient to demonstrate the utility of ACT. Log files from sites not having a zoomed-out PVD cannot, in isolation, provide sufficient information about the simulated battle to accurately determine the effects of ACT.

Of the multitude of simulation runs, three are being examined in detail: 14 November 1995 runs 1, 3, and 5 of Event 1 (WAN/LAN multicast, QO, fidelity reduction). These particular runs were selected because:

- A large fraction of the playing field was utilized.
- Use of ACT generally prevented simulations from being overloaded.
- Log files contained long-duration, stable sections.

Figure 20 shows the total DIS PDU traffic recorded by the data logger on the NRaD-A LAN. These three runs exhibit comparable levels of traffic.

5.3.2 DIS PDU Traffic

The traffic rates of different kinds of PDUs are shown in figure 21 for three different tests. Entity-state PDUs were the most common, followed by CP packets. An examination of the traffic rates of different kinds of PDUs reveals that the relative occurrences of different kinds of PDUs does not vary much as to time, thereby allowing the temporal dependence to be neglected. Figure 22 shows the results of (1) calculating the occurrence probabilities of different kinds of PDUs for 1-sec intervals, and (2) selecting the median occurrence likelihood and a spread about this range that covers the calculated values for half the intervals (i.e., 25% to 75% cumulative probabilities).

Table 7 summarizes the occurrence or likelihood of different classes of PDUs. The spread arises from (1) differences in scenarios among the runs; (2) differences in host and network performance among the runs; and (3) temporal variations in the mix of different classes of packets.

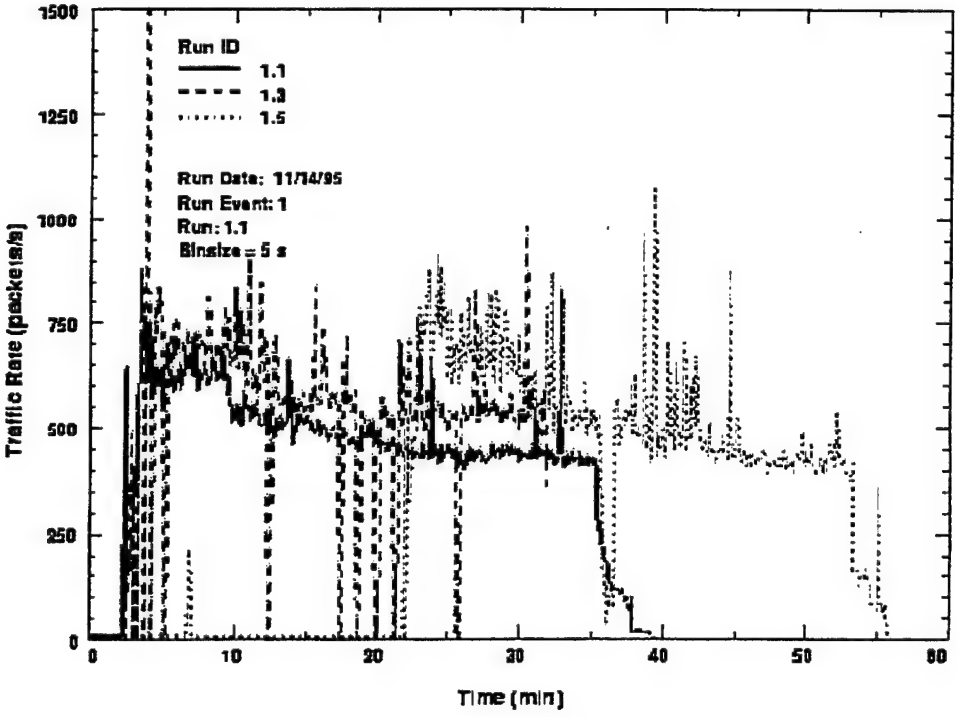


Figure 20. DIS network traffic as observed by the NRaD-A LAN for three runs of Event 1 (multicast, QO, and fidelity reduction) and recorded by the data logger.

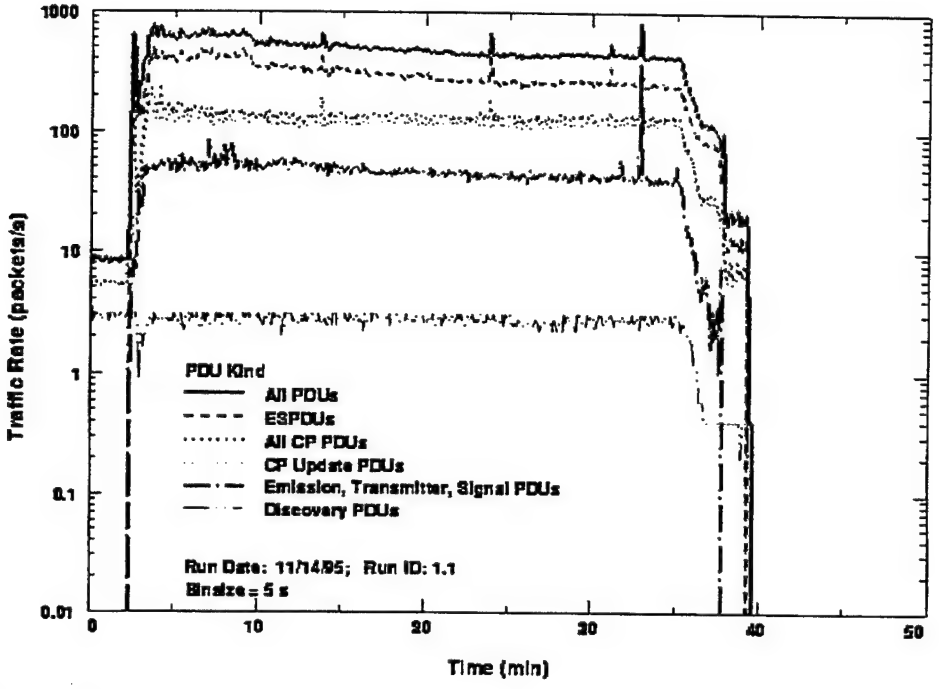


Figure 21. Traffic rate of different kinds of DIS packets as observed by the NRaD LAN for run 1 of Event 1. ESPDUs and CP PDUs are the most frequently occurring kinds of packets (N.B., traffic rate is numerated on a logarithmic scale).

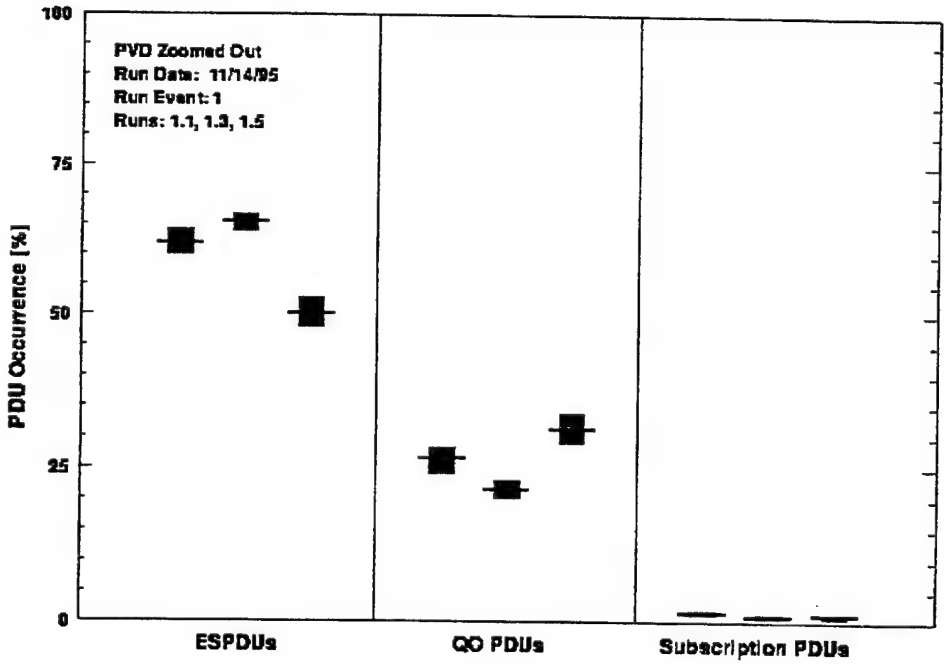


Figure 22. Occurrence likelihood for different kinds of DIS PDUs. The solid center line indicates the median value over a stable section of the exercise while the gray bar denotes a spread about this value that covers 50% of the 1-sec time intervals (25% to 75% cumulative probability).

Table 7. Likelihood of occurrence of PDU kinds.

PDU kind	Likelihood (%)
Entity State	47 to 67
QO	20 to 35
Subscription	1 to 3

The large rate of entity-state PDU traffic should have been anticipated from earlier exercises. The low rate of subscription traffic is reassuring, indicating that the incorporation of a subscription agent did not introduce a large amount of overhead. In contrast, the significant traffic from the QO packets is new, and may initially cause some concern. However, as will be shown later, these packets (which tend to be relatively small) allow a reduction in the number of entity-state PDUs that must be sent.

To better understand the dominant sources of QO packets, and how the QO protocol can be tuned to reduce network traffic, it is informative to display the distribution of different kinds of QO PDUs, as shown in figure 23. Update packets dominate, accounting for 62% to 94% of the total QO packets; these packets are sent when entities (1) transition between an active and a quiescent state, or (2) some of their characteristics change. Update packets can also be sent in response to NACKs. Stop-start motion, such as a tank turret slewing from side to side, can induce update packet transmission. These are the only QO packets that logically represent changes to the state of entities; all other

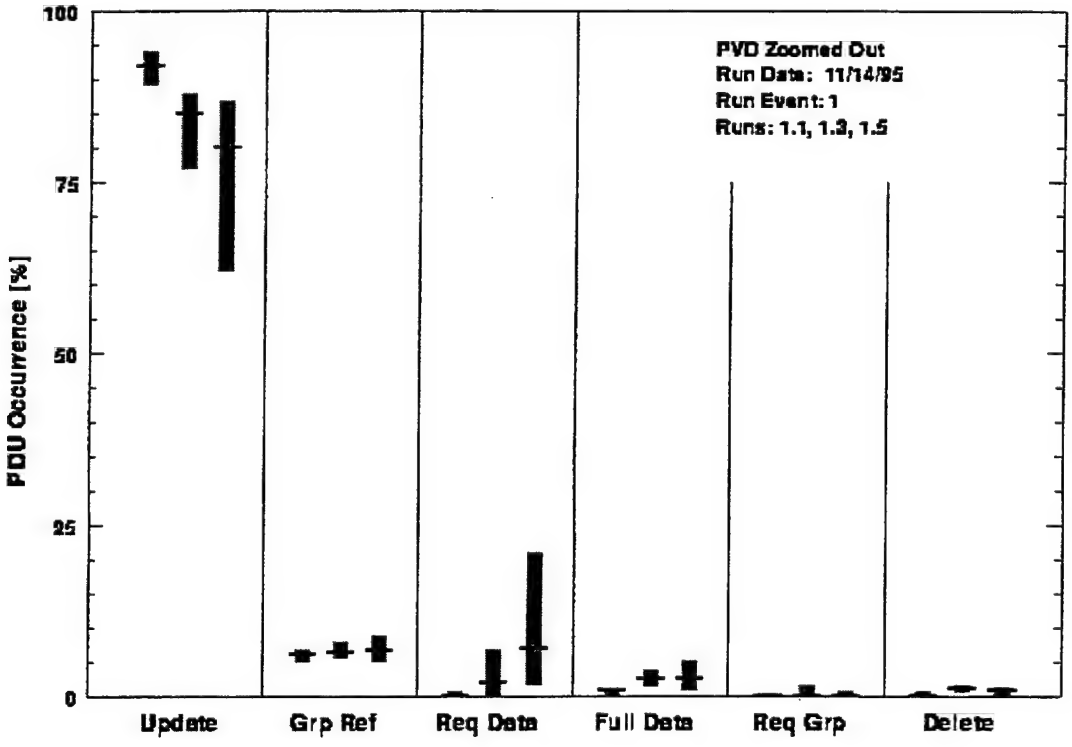


Figure 23. Breakdown of PDUs used to support QO by kind.

kinds of QO packets are used to ensure that hosts share a consistent view of the simulation. The CP update packet overhead can be reduced significantly by bundling them with entity-state PDUs since they are generated simultaneously and sent to the same multicast address. Group refresh PDUs (i.e., consistency mechanism heartbeats) account for only 5% to 10% of the QO PDUs. Request Data QO packets, also known as NACKs, represent 0% to 23% of the QO traffic, and indicate (1) faults in network connectivity or host performance, or (2) a late joiner in the exercise. The larger occurrence likelihood of Request Data QO packets during run 1.5 indicates that more difficulties were experienced in maintaining a consistent state during this run than during either run 1.1 or run 1.3, probably because the machines were on the verge of overload during run 1.5, and a larger percentage of packets were dropped. Finally, the discovery protocol produced the fewest amount of packets, almost negligible compared to the others.

5.3.3 Entity Activity

Since the PVD at the NRaD-A LAN was fully zoomed-out (observing a wide field-of-view during the simulation runs, some information concerning all entities on the playing field reached this LAN; although a large fraction of the high-fidelity traffic was not sent to this site, there is adequate information for deriving the temporal dependence of entity count, as shown in figure 24.

Figure 24 shows that there are significant differences in entity counts among the three runs, with values typified in table 8.

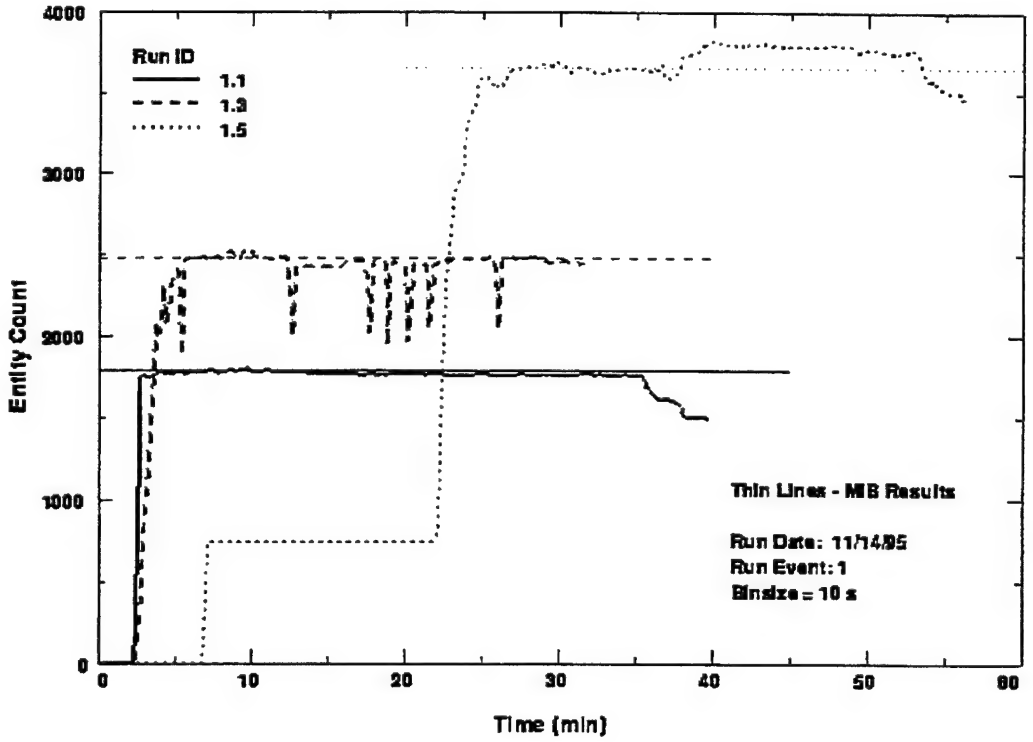


Figure 24. Total count of entities observed during three runs of Event 1. Results obtained using the log files (thick lines) are consistent with the typical maximum reported values of the corresponding MIB variable (thin lines). Note the disparity in entity count among the three exercises, even though the traffic levels were comparable.

Table 8. Entity count variations in Test Event 1.

Run ID	Entity Count (log file)	Entity Count (MIB)
1.1	1760–1800	1781
1.3	2450–2510	2476
1.5	3600–3820	3654

As should be anticipated, entity counts extracted from analysis of the NRaD-A LAN log files are consistent with the reported values of the entity count MIB variable; it should be noted that the MIB variable was time-dependent, and that the reported value represents a typical peak entity count. Finally, during Event MAX, the entity count reached 5249, the greatest amount tested together in an exercise.

Although these runs are composed of a considerable number of entities, many of them were not doing anything "interesting" at any given time. Figure 25 shows that only 10% to 40% of the entities were active at any given time, where two-thirds are quiescent on average with the relative ratio diminishing with time (i.e., as entities became disabled). Note that run 1.5, which contained more entities than the other two runs, had a smaller fraction of active entities; this may explain why the network traffic levels for the three runs are comparable even though the corresponding number of entities differs by a factor of two.

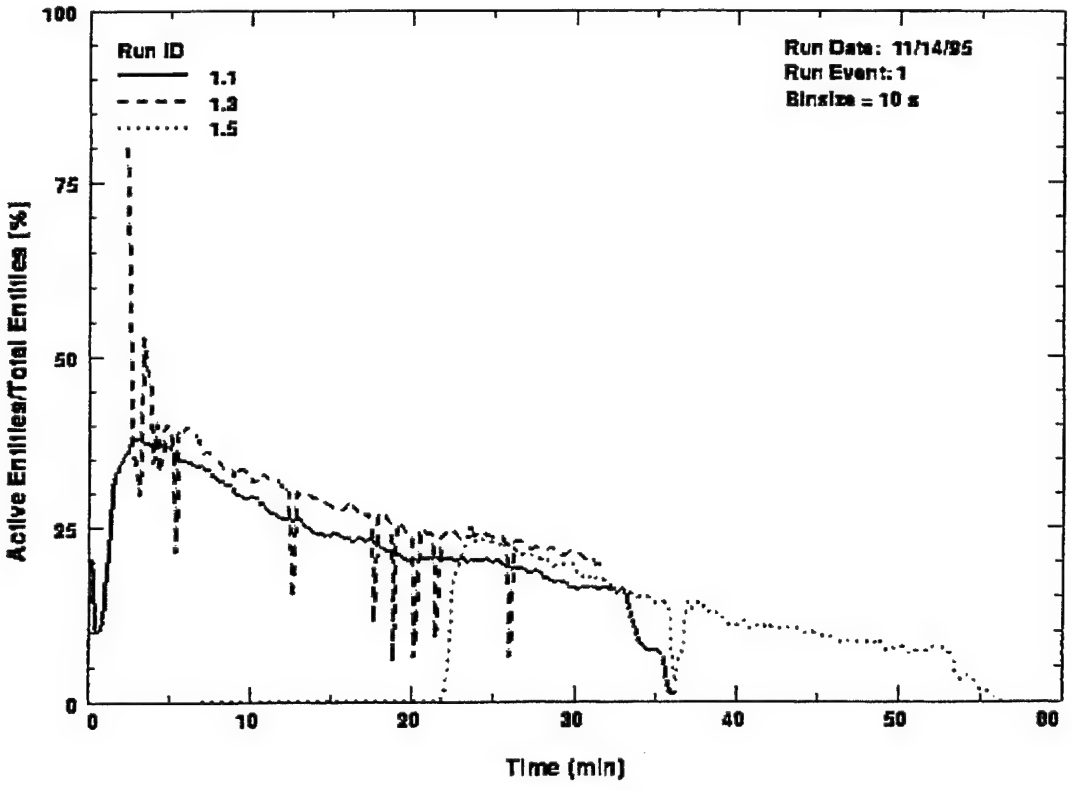


Figure 25. Fraction of entities that were considered active as a function of time. As should be expected, this fraction decreased with time as units became disabled.

5.3.4 Extrapolations

There are two basic ways to study the effectiveness of ACT:

- Execute scenarios with traffic reduction techniques enabled or disabled and compare statistics of the multiple runs;
- Extrapolate traffic levels based on logged data from a run in which all ACTs were enabled.

While the first approach is theoretically preferable, since it appears to provide objective measurements concerning the effectiveness of a subscription agent, the QO data, the consistency mechanism, and the fidelity reduction, this approach encountered difficulties because the computers involved in the runs with ACT disabled became overloaded and behaved differently than if running within their capacity. This change in behavior due to overloading implies that myriad internal parameters were not controlled (or even recorded), thereby precluding a fair comparison. The second approach, which is presented in this report, requires that analysis code be written to reconstruct the state of applications and entities involved in the simulation so that it can be rerun in logical time (not real time); an accurate extrapolation also requires that all PDUs transmitted during the run be logged for later analysis.

Since log data was not available from all sites, complete reconstruction of the state of applications and entities participating in the simulation runs is not possible; however, enough data are available to assess the benefits of ACT. Two different approaches were used to determine the effectiveness of traffic reduction techniques in the ED-1A exercise:

- PDU traffic observed by the NRaD-A LAN was studied to determine its kind, the originating application, and the destination; approximations were made to extrapolate these traffic levels to parallel scenarios in which only a subset of traffic reduction techniques were utilized.
- PDU traffic was examined to allow subscription requirements to be derived for each application; counters were maintained to record the PDU traffic observed by the hosts based on recorded traffic and calculated subscription requirements. This analysis technique allowed the team to study the effectiveness of multicast addressing.

5.3.4.1 Traffic Extrapolation Method. Note that the techniques used to extrapolate the traffic levels that would be present if the scenarios were rerun with different ACTs enabled rely on several assumptions:

- Sites have comparable mixes of entity types and activity levels of entities.
- Sites have similar amounts of overlap with entities at other sites.
- Applications at different sites experienced similar loading conditions.

Mild violations of these assumptions should not drastically affect the results, particularly if the applications on the NRaD-A LAN exhibit "average" behavior. More sophisticated analysis programs could be written that would not rely on the above assumptions, but would require additional time.

The log files are processed to count the number of different kinds of packets sent from the different hosts during each 1-sec interval. Of particular interest are quantities listed in table 9.

Table 9. Entity-type symbol definitions.

Symbol	Definition
Eq,s	Number of quiescent entities at site *s*
Hs	Number of high-fidelity ESPDUs from site *s*
Ls	Number of low-fidelity ESPDUs from site *s*
Qs	Number of multicast QO PDUs from site *s*
Qo	Number of broadcast QO PDUs
So	Number of broadcast subscription PDUs
L's	Number of low-fidelity packets from site *s* if QO were disabled

The first six quantities were counted directly from the log file; the seventh quantity was calculated by summing counts of low-fidelity ESPDUs from active entities with counts of low-fidelity ESPDUs from quiescent entities that were not in response to QO NACKs, and with simulated "heartbeats" of these quiescent entities that would be needed if QO were disabled. The actual traffic measured on the NRaD-A LAN can be expressed in terms of these quantities by

$$T[\text{PVD, multicast, QO, fidelity reduction}] = Qo + So + \sum(Hs + Ls + Qs),$$

where the sum is taken over all sites *s*.

The baseline that will be used for comparing the effects of various traffic-reduction techniques is the extrapolated amount of traffic that would have been received if the simulation were run using a broadcast communication mode with no ACT; this traffic level can be expressed in terms of the parameters listed above as

$$T[\text{broadcast}] \approx Hs' + Ls' + \sum_{s \neq s'} \left(1 + \frac{Hs'}{L's'}\right) \cdot L's,$$

where *s'* refers to the NRaD-A LAN. The rate of high-fidelity ESPDUs from remote sites is estimated by multiplying the number of low-fidelity ESPDUs observed (which provides an indication of which entities are participating in the exercise) with the ratio of high-fidelity to low-fidelity ESPDUs recorded at the NRaD-A LAN. The validity of this expression relies on entity activity at the NRaD-A LAN being comparable to corresponding activity at other sites. Note that there is no subscription traffic and no QO traffic.

5.3.4.2 Effect of Multicast Addressing. Multicast addressing can provide a significant reduction in traffic observed on a LAN if the regions of interest viewed by applications on the LAN do not significantly cover the overall playing field. For this reason, multicast addressing without any other traffic reduction technique provides no reduction in traffic levels on the LAN when there is a zoomed-out PVD present; in fact, multicast addressing increases traffic levels slightly because of the addition of subscription traffic (however, even with a PVD present, it does allow traffic reduction at the application level). An approximation to the network traffic that would be observed if multicast addressing were used is given as

$$T[\text{PVD, multicast}] \approx Hs' + L's' + So + \sum_{s \neq s'} \left(1 + \frac{Hs'}{L's'}\right) \cdot L's .$$

This expression is identical to the one for broadcast traffic with the addition of subscription traffic.

If there were no sensor such as a zoomed-out PVD at any application on the LAN, the network traffic could be estimated as

$$T[\text{no PVD, multicast}] \approx Hs' + So + \sum_{s \neq s'} \left(1 + \frac{L's'}{Hs'}\right) \cdot Hs .$$

High-fidelity ESPDU traffic from remote sites provides an indication of which entities would be seen if a PVD was not present. The ratio of low-fidelity traffic to high-fidelity traffic at the site is used to estimate how much traffic would be received on the low-fidelity channel. As machines become loaded beyond their capacity, they tend to "tick" more slowly and, therefore, generate less traffic on the high-fidelity channel; as this happens, the ratio of low-fidelity to high-fidelity traffic becomes large and unstable, and the above equation becomes less reliable. As logged data from LANs with no WAVs becomes available, extrapolations based on the formula presented above should be replaced.

5.3.4.3 Effect of Fidelity Reduction. Fidelity reduction would only have an effect on DIS traffic observed on a LAN if there were a wide-area sensor that only required low-fidelity data from entities located over an extended geographic region, such as a zoomed-out PVD or a high-altitude aircraft. The following formula provides an estimate to the traffic that would be observed on a LAN if there were a zoomed-out PVD, and both multicast addressing and fidelity reduction were used:

$$T[\text{PVD, multicast, fidelity reduction}] \approx So + \sum (Hs + L's) .$$

5.3.4.4 Effect of Quiescent Entity Suppression. The CP reduces the number of ESPDUs that must be sent, but requires a certain amount of overhead. If QO were used with the system operating in a broadcast mode, the network traffic could be approximated as

$$T[\text{broadcast, QO}] \approx Hs' + L's' + Qs' + Qo + \sum_{s \neq s'} \left[\left(\frac{Hs'}{L's'}\right) \cdot L's + Ls + Qs \right] .$$

Network traffic anticipated if QO were used in a multicast environment with a WAV present is equivalent to the preceding equation with the addition of subscription traffic.

Network traffic anticipated if QO were used in a multicast environment with no PVD present is given by

$$T[\text{no PVD, multicast, QO}] \approx Hs' + Ls' + \beta s' Qs' + So + Qo + \sum_{s \neq s'} \left[\left(1 + \frac{Ls'}{Hs'}\right) \cdot Hs + \beta s Qs \right] ,$$

$$\beta s \equiv \langle \text{ overlap } \rangle \cdot 1 + \left(1 - \langle \text{ overlap } \rangle\right) \cdot \frac{Eq,s}{\sum Eq,s''} ,$$

where βs is used to scale the QO traffic with the degree of scaling determined by the number of quiescent entities at the different sites, and the degree to which entities residing at different sites overlap (based on a visual inspection of the entity laydown, it was assumed that $\langle$ overlap $\rangle = 0.25$). As the overlap approaches unity, βs also approaches unity (i.e., QO traffic is not reduced by removing the WAV); as the overlap approached zero, QO traffic scales by the fraction of quiescent entities that are resident at the site.

5.3.4.5 Comparison of Estimated Traffic Levels. Figure 26 shows the total DIS PDU traffic levels anticipated if the exercises were rerun with different combinations of traffic-reduction techniques, the machines did not become overloaded, and a PVD was fully zoomed out (N.B., based on data from the NRaD-A LAN). Fidelity reduction provides a modest decrease in traffic levels, but not as much benefit as QO; the combination of these techniques provides even more benefit.

The occurrence of various levels of reduction in traffic levels (compared with broadcast) is shown in figure 27; of particular interest are the median level of reduction (i.e., at least this much reduction was found to occur in half of the 10-sec intervals examined), and the spread about this value that describes the reduction levels in half of the intervals (cumulative occurrence probabilities between 25% and 75%).

Traffic-reduction levels for the case that a zoomed-out PVD is resident at the site are derived from figures such as figure 27 for runs 1.1, 1.3, and 1.5 of Event 1, as shown in figure 28 and summarized in table 10.

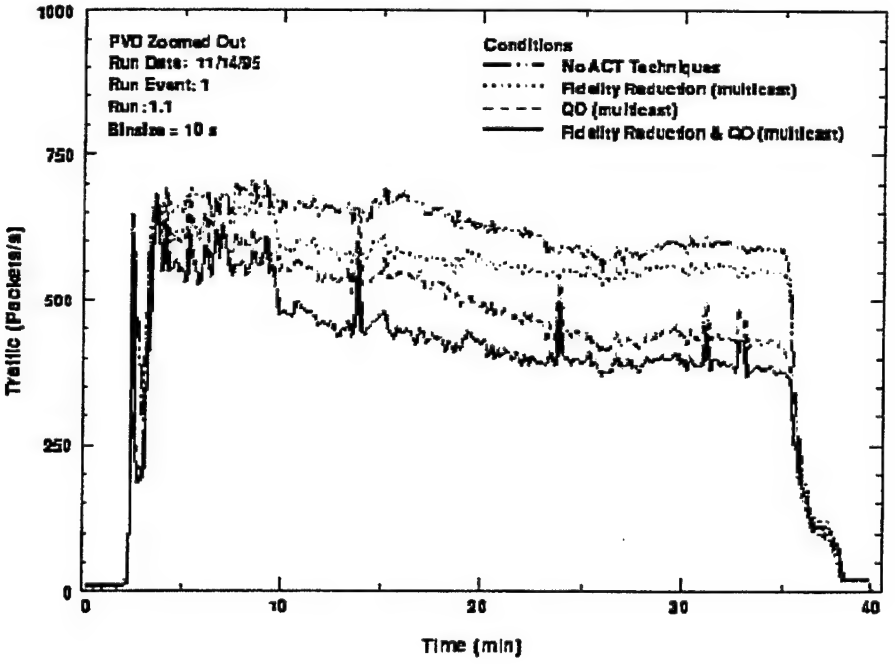


Figure 26. Estimates of network traffic level at a LAN having a fully zoomed-out PVD as a function of time and traffic-reduction technique. Back-calculation techniques were used to extrapolate these traffic levels from measurements during which all reduction techniques were utilized (solid curve).

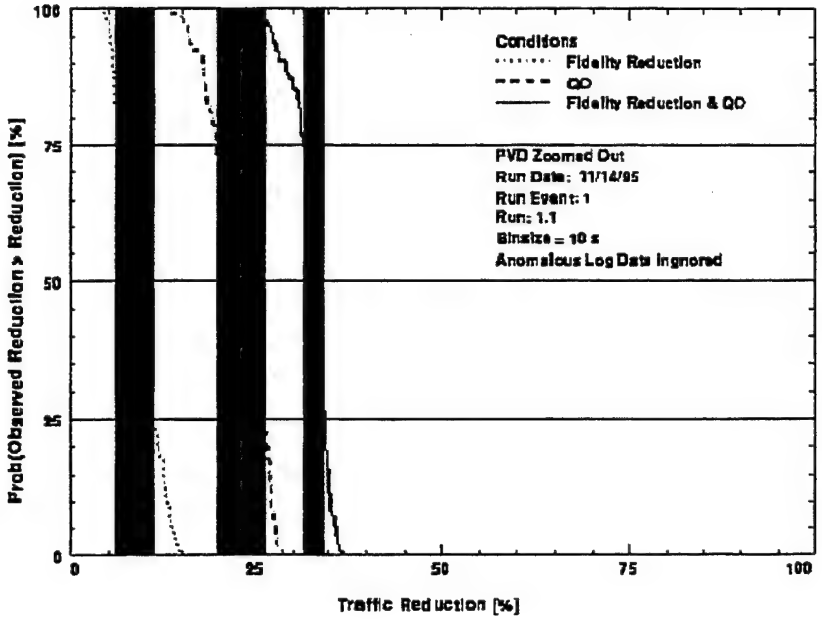


Figure 27. The cumulative probability of achieving various reductions in the LAN traffic of DIS PDUs (as compared to using no reduction techniques) for three combinations of reduction techniques. The solid vertical lines indicate the median level of traffic reduction, while the shaded gray regions denote the spread about this value (25% to 75% cumulative probabilities).

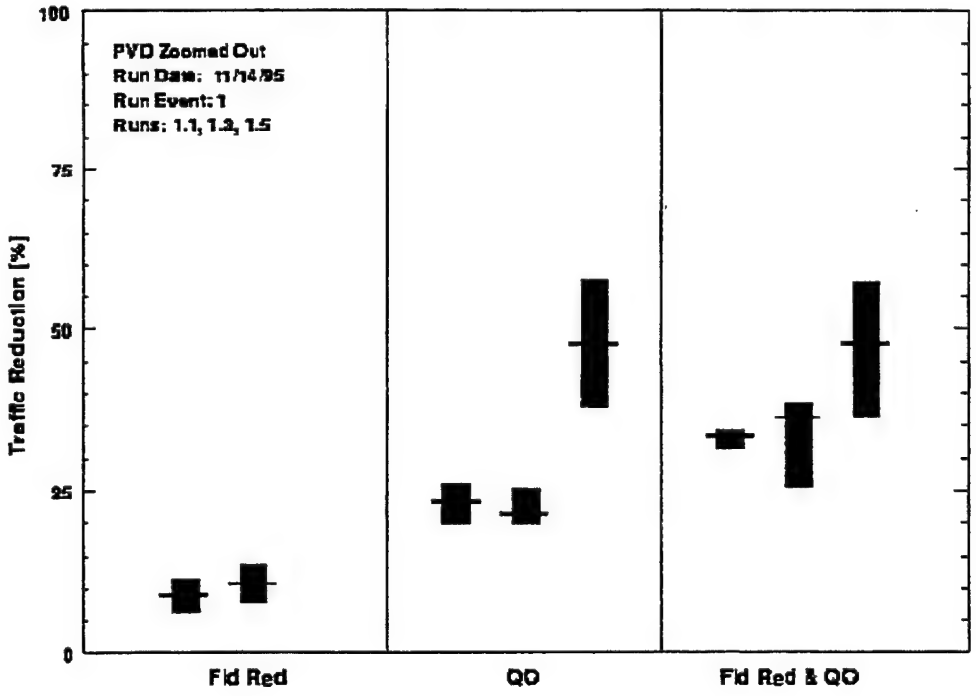


Figure 28. Median levels of LAN traffic reduction and spreads around these levels for three combinations of reduction techniques and three runs of Event 1. These results are specific to a case in which there is a zoomed-out PVD on the LAN.

Table 10. Traffic-level reductions for various ACTs with PVD.

Technique	Run 1.1 (%)	Run 1.3 (%)	Run 1.5 (%)
Fidelity reduction	6 to 12	8 to 14	<1
QO	19 to 26	20 to 25	38 to 57
Fidelity reduction & QO	31 to 34	25 to 38	36 to 57

Fidelity reduction provides a noticeable traffic reduction for runs 1.1 and 1.3, but not for run 1.5. This disparity may be due to the heavy loading of run 1.5, which tends to reduce the amount of traffic on the high-fidelity channel; the traffic estimates for this run are stable since traffic from the high-fidelity channel enters into numerators of ratios. QO provides significant benefit in all three runs. The benefits of QO and fidelity reduction are approximately additive. The effect of multicast addressing is not presented, since it provides no traffic reduction in LAN load when there is a zoomed-out PVD at the site.

Similar calculations for the case that there is no zoomed-out PVD at the site are shown in figure 29 and summarized in table 11.

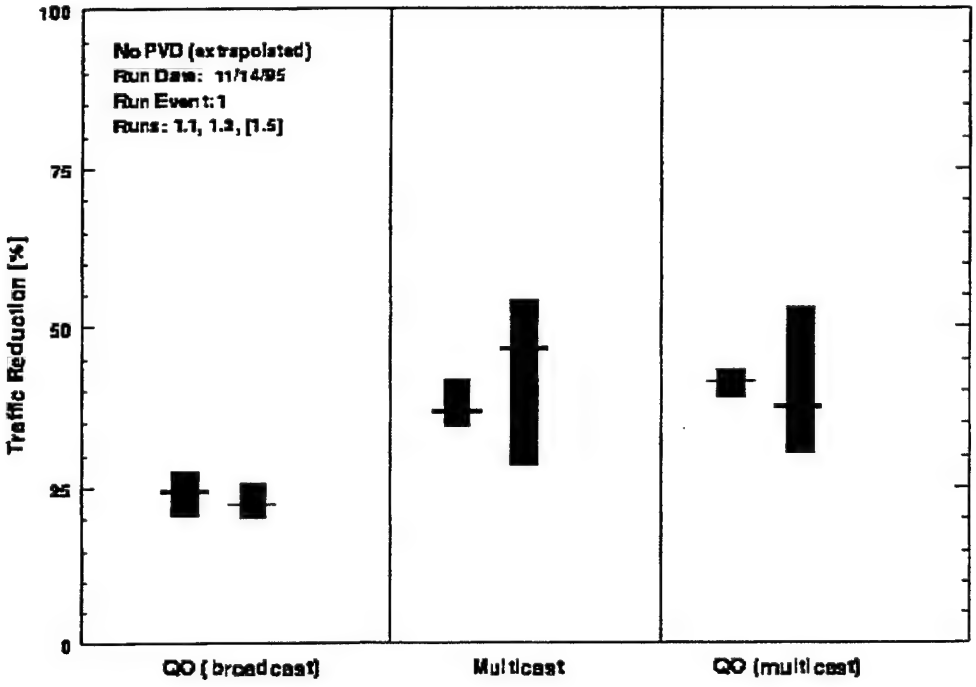


Figure 29. Effectiveness of QO and multicast towards reducing network traffic if there is no zoomed-out PVD on the LAN. The extrapolation techniques applied to data collected from the NRaD-A LAN (which had a zoomed-out PVD) may not be reliable for estimating the reduction levels for sites not having an active PVD.

Table 11. Traffic-level reductions for various ACTs with no PVD.

Technique	Run 1.1 (%)	Run 1.3 (%)	Run 1.5
QO (broadcast)	20 to 27	20 to 26	*
multicast	34 to 42	29 to 56	*
QO (multicast)	39 to 43 †	30 to 53	*

†These preliminary results are based on less-reliable extrapolations and should not be taken as definitive; results based on analysis of log data from LANs without a resident WAV should replace these reduction values.

*Run 1.5 is not included since the extrapolation formulas presented above did not produce "stable" levels of traffic reduction, probably due to a relatively small number of high-fidelity entity-state PDUs recorded during certain segments of the exercise run. It should be stressed that the extrapolation formulas used for estimating the network traffic that would be observed at a site without a PVD are not as reliable as the formulas for estimating traffic if a zoomed-out PVD was present. These results imply that QO or multicast addressing provide a substantial reduction in network traffic, but that a combination of these techniques may not substantially reduce traffic levels. So far, there have not been any fundamental explanations behind the inability of QO with multicast addressing to provide little traffic reduction beyond multicast alone; it is conceivable that this indicates limitations of the extrapolation formulas for cases not involving a zoomed-out PVD.

5.4 OVERLOAD MANAGEMENT

Of the two major elements of the OM algorithm, the team implemented only the LL aspect of the algorithm. Due to an artificially high threshold setting, OM was not invoked during ED-1A. During pre-demonstration testing, LL was effective in maintaining traffic levels below the set threshold and, it is assumed, preserved exercise validity better through priority-based rather than random dropping of packets.

5.5 APPLICATION TRANSLATOR

The AT successfully allowed a legacy (DIS 2.03) simulator and LAN to participate in a DIS 3.X exercise. The AT is a bridge between the two technologies, allowing entities that use both protocols to effectively engage each other throughout the exercise. A median decrease of 52.8% was observed in the packet traffic transmitted from the legacy LAN to the DIS 3.X LAN. This was largely due to the AT's suppression of quiescent entity heartbeat PDUs from the legacy LAN. While packet traffic transmitted from the DIS 3.X LAN to the legacy LAN increased by 236%, 82% was self-generated by the AT in the regeneration of quiescent entity heartbeat PDUs. A maximum entity count of 796 entities was observed by the AH, with 44 entities (an M1 Battalion) generated on the 2.X ModSAF.

5.6 TIME SYNCHRONIZATION

No direct time synchronization data were gathered. Analysis of time synchronization from the data collected indicates several interesting things about time synchronization performance.

First, the SNMP manager workstation "warbird" was not synchronized at all. This host appeared to be 15 minutes off most of the time. As a result, MIB data gathered at warbird cannot be finely correlated with MIB data from "pacers," the other management station, or with trace data.

Second, the WAN data logger at NRaD had significant synchronization errors. One-way trip times computed between various HPAGs and this data logger showed drift and discontinuities of 10 msec or so. It is believed that this is due to a bug in the SGI operating system. Discounting these discontinuities and accounting for resulting synchronization problems, the data seem consistent with the one-way delays computed by the bi-level protocol. Further, short-term delay variance seems to be small, on the order of 2 to 5 msec. Very little delay variance can be attributed to the WAN, as the delay variance from one NRaD site to the other is not easily distinguishable from the delay variance from the other sites. The time synchronization errors make a rigorous version of this analysis very difficult.

Next, the HPAGs achieved very good synchronization. Time synchronization in ED-1A was more complex than simply using off-the-shelf solutions. A version of NTP was produced that used the cycle counter in the SGI workstations to obtain more reliable time. This was successful on the HPAG platform. In order to have "free off-the-shelf" time synchronization software meet future needs, vendor implementations of in-kernel time-keeping will need verification. As errors can be introduced by new features, this verification must be done for every new operating system release and hardware combination. One-way trip times between HPAGs were very constant, as discussed above. Some packets encountered larger delays, but there were almost none encountering lower delays. Thus, it is concluded that clock synchronization of HPAGs was very good. Additional refinement is required in the area of configuration support and verification. Experts verified and configured the time synchronization system. As a result, high-quality synchronization was achieved for many hosts. The process was neither "cookbook" nor automated, however, meaning that future results could vary without further effort to remove all opportunity for operator error.

Finally, the simulator's use of the synchronized clocks appeared to be correct. There were no reports on the teleconference of ModSAF complaining about incorrect time stamps during any of the tests. There were many reports of other ModSAF diagnostic messages. ModSAF should have reported any time stamp errors if they occurred.

5.7 QUALITY OF SERVICE

QoS support was not implemented for ED-1A. Neither the IP WAN nor the underlying ATM network could provide QoS support. However, data were collected during ED-1A will be very helpful in designing future QoS support. In this section, the nature of this data is discussed and analyzed. Some work was done to make existing RSVP signaling code (the "ISI" implementation of RSVPD) work on the SGI platform, as well as to allow making a large number of reservations from a single program.

In addition to mechanisms to make and enforce reservations, a hard problem is deciding which reservations to make. In the classical DSI model of simulation communication, all traffic from each site is transmitted to all other sites. A static reservation is made (via a Stream Transport Two (ST2) stream) from each site to all other sites. The size of this reservation is an estimate of the total traffic from the site. This model is workable and reasonably straightforward because (1) The total traffic generated by a collection of entities is relatively easy to predict, and (2) the set of destinations to which this traffic must be delivered is predetermined.

Given the use of multicast as a traffic reduction technique, the question of which reservations to make is far more complex. With multicast, data are delivered only where needed, and this cannot be fully known in advance. Now the amount of bandwidth needed from site A to site B is not a function of the traffic being locally generated at A, but of the amount of traffic generated at A that is needed

at B. This is more complex because (1) it is not at all clear before an exercise which part of A's traffic B will need, and (2) this fraction is likely to change during the exercise.

The data showing communication patterns among sites show that there were, in fact, changes in the relative amount of traffic sent between sites, as shown by the figures in Appendix B. While this is not surprising, it does indicate that simple static reservations may not be adequate. This implies that either over-provisioned static reservations or dynamic reservations will be required in the future. While only 1-minute averages were examined, it was found that generally there were not wild variations from minute to minute. This is encouraging, indicating that ATM reservations based on traffic needs may be possible in the future.

5.8 NETWORK IMPLEMENTATION

The network and system achieved the 5000-entity goal during the Event MAX test with 5249 entities over 7 sites and 62 workstations. This is one of the highest active combat entity engagements ever performed with distributed simulation. It required that no sites except for NRaD-A use a PVC to avoid saturating the network with all packets to all sites.

The implementation of the network included many components that were required to work in unison, but the analysis results are documented separately. The discussion below will be divided into four areas: the WAN, the LAN, the Routers, and Data Collection via SNMP.

5.8.1 The WAN

The ATM WANs were substantially over-provisioned for the traffic loads offered by ED-1A. Aggregate WAN loading by the ED-1A simulations did not exceed the 5 Mbps average, which is considerably less that the 45 Mbps available in AAI and the 2.4 Gbps in ATDnet. Bandwidth reservation was neither needed nor attempted. In STOW 97, with larger scenarios and bandwidth demands as well as other competing users on the network, QoS metrics that deliver appropriate service will be needed.

The ATM networks delivered very high bandwidth and consistent low latency. Typical host-to-host latency across the combined LANs and WANs were as follows:

1. West Coast (NRaD) to East Coast (TEC, IDA, NRL, DARPA): 65 to 70 msec.
2. West Coast to Texas: 75 to 80 msec.
3. Texas to East Coast: 50 to 55 msec.
4. Across multiple switches in ATDnet (e.g. NRL to TEC): 4 to 6 msec.
5. Across two switches in ATDnet (e.g. TEC to IDA): 1 to 3 msec.

5.8.2 The LAN

In pre-exercise testing, the Sun workstations were selected as the multicast traffic sources so that sufficient levels of traffic to saturate the ethernet on large numbers of multicast groups could be generated. When traffic generation was attempted with one Sun and one SGI workstation, the SGI workstation tended to back off ethernet transmissions at high-traffic rates while the Sun dominated the load put on the network. The SGI workstation's apparent excessive sensitivity to colliding with other traffic on the ethernet reduced its effectiveness as a traffic generator. It is possible that the SGI ethernet board may be designed with a modified back-off algorithm to avoid the "capture effect" in

typical high-performance server/low-performance client(s) scenarios. While this configuration may prove beneficial in typical client/server environments, this causes the SGI machine to "lose its edge" in competitively loading an ethernet with connectionless packet traffic at high congestion levels. Further, when simulation applications running on SGI workstations are heavily loading a node-site LAN, the back-off characteristic observed above could result in erratic workstation behavior and degraded application performance.

When two Sun SPARCs were used, they could reach an even balance of transmission at high packet rates and load the LAN segment with up to a total of 7000 pps (64-byte packets). As a result, the Suns were used for traffic generation while the SGI workstations were used for reception.

ED-1A simulation traffic on the node-site LANs resulted in consistent loading above 3 Mbps. This loading level was saturating the 10-Mbps shared ethernet. Due to ethernet limitations, the workstations were benchmarked to the FDDI interface.

5.8.2.1 Silicon Graphics Indigo2 with Ethernet. The SGI workstation displayed a sensitivity to ethernet congestion. When an SGI attempted to load the ethernet at high packet rates while competing with other workstations also injecting high-traffic loads on the same ethernet segment, the SGI would stop transmitting packets. Monitoring with a sniffer revealed that the SGI workstation squelched its transmissions, allowing other workstations to "capture" the LAN. With multiple SGI workstations on the LAN, very high ethernet utilization numbers were not attainable with the small packet sizes.

5.8.2.2 Silicon Graphics Indigo2 with FDDI. The MGEN tool set was used to evaluate the multicast transmit and receive capabilities of the SGI Indigo2 configured with a FDDI interface. The test bed was configured with one transmit SGI, one receive SGI, and a Networks General FDDI sniffer, all attached to a single ring. The sniffer verified packet transmission when traffic volume exceeded the receive capability of the MGEN tool set. Point-to-point unicast testing confirmed connectivity. The transmitting station could pass 10,000 pps onto the FDDI LAN. This number was verified by the sniffer. The transmit rate was unaffected by the number of multicast groups used. On the receive side, there was some anomalous behavior. Some initial runs indicated that the receiver could receive 1000 pps over 100 multicast groups, and even 500 pps over 1000 groups, but these tests could not be repeated after initial success. Beyond these initial runs, the limit for the number of groups that could receive data was 31. The limit of 31 groups appears to be independent of the transmitted data rate, with data rates of 10 to 10,000 pps, tested. The groups that received data were spread across the whole range of chosen groups and had perfect reception of the packets transmitted to them. The machines were rebooted and tests were run again with the same results. The specific groups that passed data were the same every time, even after rebooting the workstations.

5.8.2.3 Sun SparcStation 20 with FDDI. The MGEN tool set was used to evaluate the multicast transmit and receive capabilities of a Sun SPARC 20 configured with an FDDI interface. A Networks General FDDI sniffer was used to verify packet transmission to the LAN. First a point-to-point (unicast) test was performed to verify packet transmission and reception. A single Sun SPARC 20 could load the FDDI segment with up to approximately 9300 pps using FGEN. Reception of unicast packets by DREC with no errors noted was verified up to the logging limits of the machine. Then a single multicast group was used for testing. While the transmitting Sun could successfully generate packets from moderate to very high rates with all expected packets observed with an FDDI sniffer, the receiving Sun workstation dropped approximately 25% of the packets even for transmission rates as low as 1 pps. This 25% loss of received packets on the FDDI interface held true for dif-

ferent numbers of simultaneously active multicast groups. When multiple groups were used, the packet loss was uniformly spread across the groups.

While the transmitting Sun could transmit to arbitrary numbers multicast groups, the receiving Sun could only successfully receive traffic on up to 14 multicast groups (even when more than 14 groups were joined by the application software). For example, when 20 groups were joined by the receiver, the first 14 groups received traffic (with the 25% packet loss) while no packets were received by the DREC application for the remaining six groups.

5.8.3 The Routers

It was anticipated that difficulties might be encountered as the hosts and routers were required to deal with high packet volumes on large numbers of simultaneously active multicast groups. The team conducted tests to find the "break points" of the various implementations of multicast in the routers and hosts with different numbers of multicast groups. Once break points were found, the router implementations were further characterized for how well they continued to perform when pushed beyond the bounds of flawless operation. Figure 30 shows an example of how the operational area was defined for router multicast forwarding performance.

When the number of active groups was small, the routers could maintain higher multicast forwarding performance in terms of pps without any packet loss. As the number of active groups was increased, the routers exhibited different patterns of packet loss. The line on the graph represents the boundary at which the router was able to forward packets without loss. Thus, the area below the line (no packet loss) is termed the "operational area," while the area above the line represents some type of degraded performance for the router in terms of packet loss. In some cases, this degraded performance was catastrophic (almost total), in other cases, degraded performance was minimal (1% to 5% loss).

5.8.3.1 The Cisco 7000 Router. The Cisco 7000 router, running Internet Operating System (IOS) 10.2(5.5) could not reliably forward large numbers of multicast packets with even a small number of multicast groups, and was incapable of passing small data loads with a large number of multicast groups. As shown in in figure 31 and table 12, the operational area for multicast is far below the normal ability of the router to forward unicast packets.

Above the maximum reliable throughput, there is no gradual degradation of performance. The failure is catastrophic. Data transmitted to some groups are not forwarded, and data are delivered partially to others. No group will receive all the data sent to it. There is no pattern to the loss of groups, and the exact results of a given run are nonrepeatable. This small operational area and indeterminate failure mode made this hardware/software combination unusable for ED-1A.

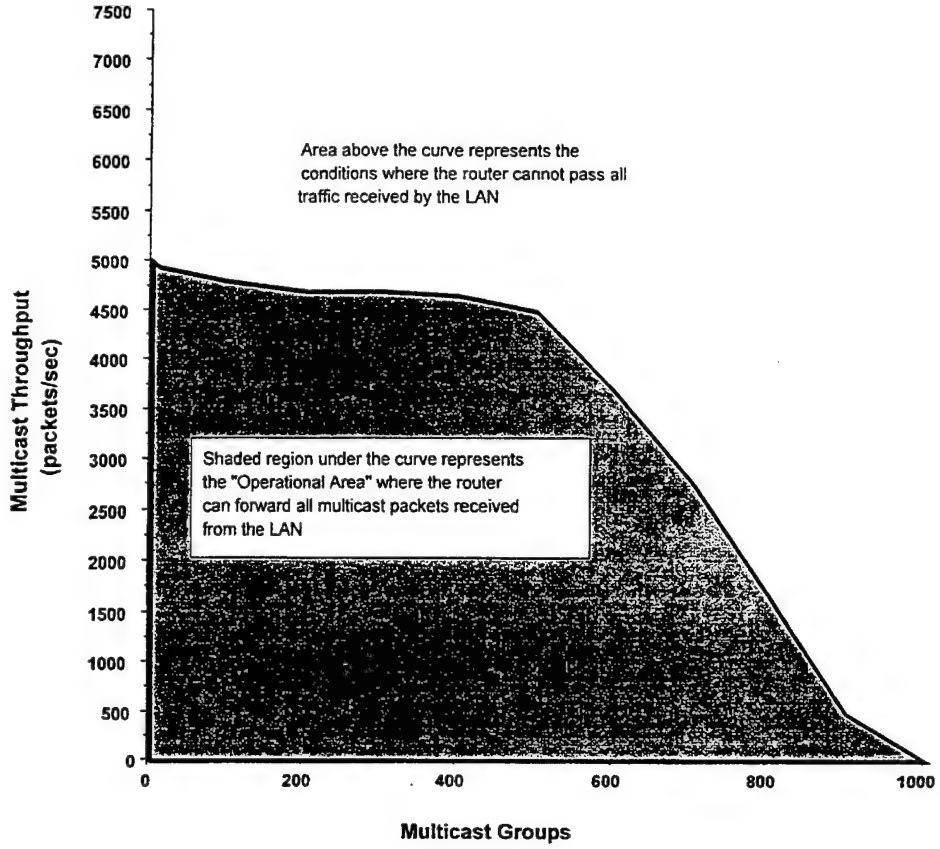


Figure 30. Router operational area.

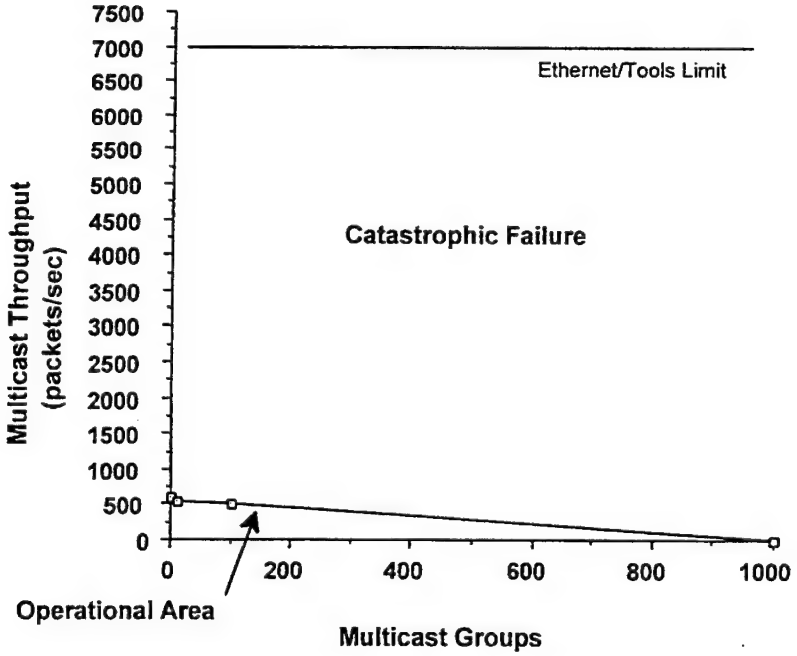


Figure 31. Cisco 7000 10.2(5.5) multicast forwarding performance.

Table 12. Cisco 7000 10.2(5.5) multicast forwarding performance data.

Multicast Groups	Maximum Reliable Throughput (pps)
1	600
10	550
100	500
1000	0

The Cisco 7000 router was next loaded with the 11.0(3.3) version of the Cisco Internet Operating System (IOS), which was first available as a commercial release in October 1995. This update includes changes to accelerate multicast forwarding performance by placing the routines in a faster forwarding path of the router operating system. The update significantly improved multicast forwarding capability, as shown in figure 32 and table 13. As expected, the router was able to forward unicast IP traffic at the maximum speed supported by a single ethernet segment. It is important to note that when driven with packet transmission rates above the break points listed, the router could forward packets and lose only approximately 2.5% of the transmitted packets. The distribution of the missing packets was uniform over the joined multicast groups.

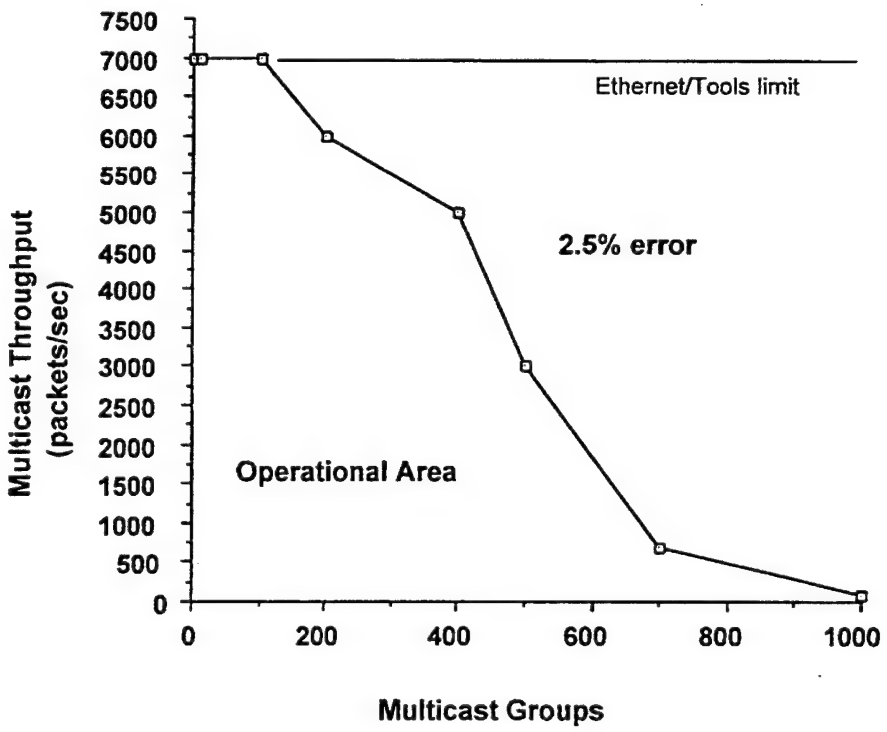


Figure 32. Cisco 7000 11.0(3.3) multicast forwarding performance.

Table 13. Cisco 7000 11.0(3.3) multicast forwarding performance data.

Multicast Groups	Maximum Reliable Throughput (pps)
1	7000
10	7000
100	7000
200	6000
400	5000
500	3000
700	700
1000	100

5.8.3.2 The Bay Networks 72000 Router. The Bay Networks 72000 BLN-2 running Version 8.11/fix2 performed well across a large range of multicast groups. As shown in figure 33 and table14, the BLN-2 could forward 6000 pps or greater to 1000 simultaneous multicast groups.

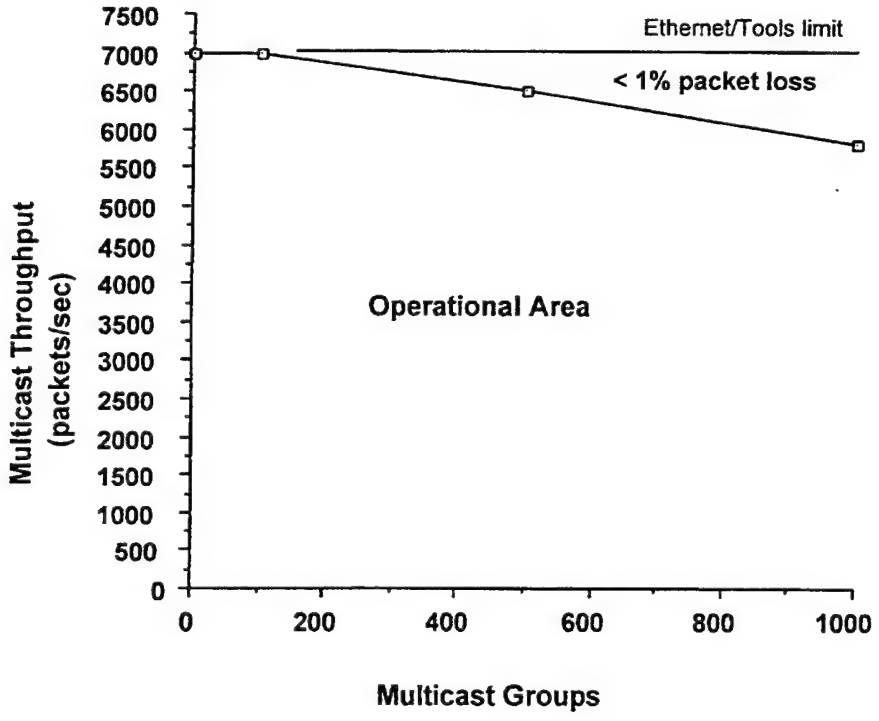


Figure 33. Bay Networks 8.11/2 multicast forwarding performance.

Table 14. Bay Networks 8.11/2 multicast forwarding performance data.

Multicast Groups	Maximum Reliable Throughput (pps)
1	7000
100	7000
500	6500
1000	6000

Above the operational area in figure 14, there is a graceful degradation up to the ethernet/tools limit of 7000 pps. When loaded above the operational area, most packets were delivered. Less than 1% of the packets offered at the transmit LAN were lost, and the loss was spread in a uniform distribution across the active multicast groups.

One observed anomaly with the BLN 72000 was a long join latency in the router. When large numbers of multicast groups (> 400) were joined by a directly attached host, the router would accept all of the group join messages, but would not fully register them until a large amount of time had elapsed (> 1 min). The router will not pass data on these groups before this long registration time has elapsed. By checking the router's MIB, note that the router will write all groups to MIB variables immediately, but will not provide information about those groups because it has not fully processed all of them.

By forcing data at the router on all groups, the registration speed for a large number of groups can be increased. If data are forced over all groups, halted, then sent again, the router will pass data on all groups.

5.8.3.3 The HPAG. The team also evaluated the HPAG's ability to forward multicast packets (much like a router). Tests were conducted to determine the HPAG's performance capabilities in its LAN-to-WAN multicast encapsulation and forwarding mode. One WAN multicast group was used. The tests were only conducted in this one direction. Future tests will be conducted to evaluate its WAN-to-LAN performance, and the HPAG's ability to perform both functions simultaneously. The results of the LAN-to-WAN testing are shown in figure 34 and table 15.

Note that the HPAG used in ED-1A was a proof-of-concept prototype that was running in user space on an SGI Indigo2, R4400 running at 200 MHz. A reference implementation of the HPAG is planned for the next "ED" series demonstration that will be optimized to run in kernel space.

The HPAG's performance capabilities were consistent across the range of tested multicast groups. The HPAG could forward approximately 1200 pps. When pushed above the limits shown in table 15, it still forwarded at the limit level (i.e., if 2400 pps were generated, the HPAG would drop approximately 50% of the packets). It is anticipated that the HPAG will suffer degraded LAN-to-WAN forwarding performance when it is simultaneously forwarding packets from the WAN to the LAN. Multiple multicast groups on the WAN may affect its forwarding performance as well. Finally, the 1200-pps limit will not scale very well when observing figures 14 through 17. The LAN packet loads are approaching the HPAG limit. Depending upon the scenario, the LAN traffic of that magnitude could easily have to pass through the HPAG. As a result, a combination of improvement in hardware and software optimization are necessary to increase the packet limit of the HPAG.

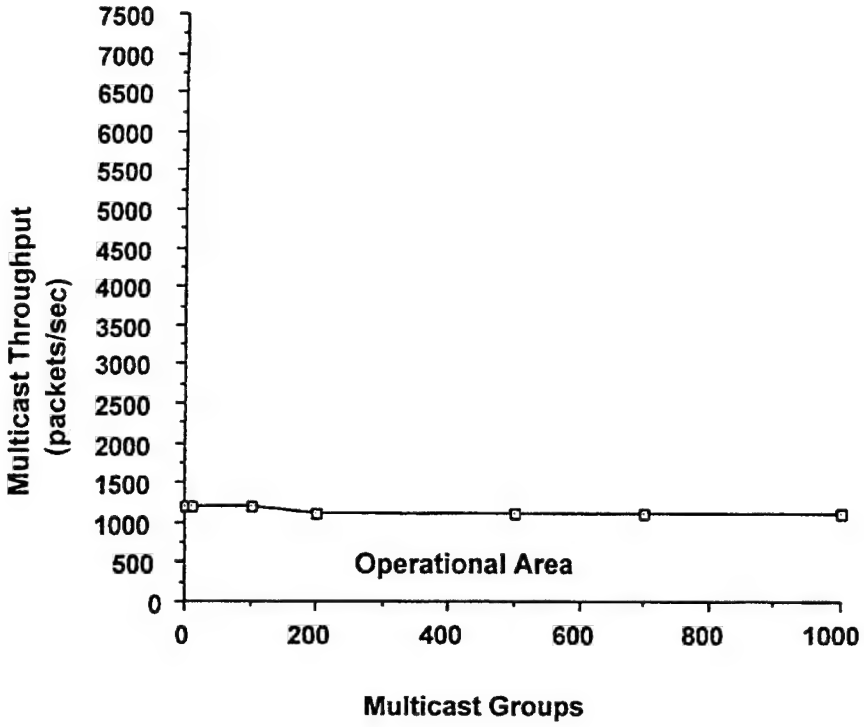


Figure 34. HPAG multicast forwarding performance.

Table 15. HPAG multicast forwarding performance data.

Multicast Groups	Maximum Reliable Throughput (pps)
1	1200
10	1200
100	1200
200	1100
500	1100
700	1100
1000	1100

5.8.4 Data Collection via SNMP

The ACTS SNMP interface performed satisfactorily during ED-1A. From a centralized location, performance statistics were successfully collected from all the ACT components at all seven sites. ACT components were also controlled via SNMP (by changing control variable values), though this functionality was seldom used.

Some problems were experienced during data collection and analysis:

- ACT SNMP subagent processes failed to respond to query for statistics when the host processor was overloaded. The SNMP subagent was assigned a lower priority than most of the other

processes on the host, and it did not execute when the host processor was busy handling higher priority threads.

- Some data collection packets were missing, resulting in gaps in data collection. Since SNMP protocol uses UDP packets, the queries or replies will be lost if the network is overloaded, resulting in gaps in data collection.
- Time mismatch between the time the data were actually collected from the ACT software component and the time logged by the data collector (HP OpenView). In ED-1A, the time data logged were used to analyze the data that do not account for the network delay in collecting the data.
- Post-processing is needed after collection. In ED-1A, it was observed that some data processing is still needed after the collection to properly analyze the data.

6. SUMMARY

Overall, the prototype communication architecture developed for ED-1A was validated. The system supported 5249 entities and met the goal of the ED-1A entity count. This section provides conclusions, observations, inferences, and extrapolations based on the ED-1A experience. The topics are arranged by ACT function.

6.1 MULTICASTING

The multicast findings are divided into three categories: WAN multicasting, ED-1A Implementation of Bi-level Multicasting, and Application-Level Multicasting.

6.1.1 WAN Multicasting

Multicasting across the WAN determined how much traffic was received by each site. In the full implementation of multicasting (an optimum number of LAN groups and 127 WAN groups), the team found the following:

1. Most simulation traffic was requested by at least one other site. Approximately 80% of all local traffic was transmitted to the WAN, so bandwidth savings on the WAN was 20% or less compared to a broadcast-based exercise. This result may have been largely due to the use of PVDs that requested traffic from virtually all areas of the simulation battlespace.
2. Virtually all WAN traffic received at a site was delivered to the LAN. This means that multicasting was functioning properly, and that unnecessary mail was not being delivered to any of the sites.
3. An overall "reduction factor" due to multicasting is not possible. By site, however, multicasting resulted in the receipt of only 35% to 60% of the traffic that would have been received under a broadcast delivery model. (This finding held true for sites not using a PVD.)
4. The benefit of multicasting for the WAN can be defeated by a distribution of forces in the virtual battlespace such that all nodes need to see every other node.
5. Conversely, it should be possible, with proper planning as to the distribution of forces, to conduct exercises with sites having dissimilar bandwidth connections.

6.1.2 ED-1A Implementation of Bi-level Multicast

1. Very little unexpected control traffic was observed, indicating that the bi-level protocol worked well and recovery mechanisms were rarely invoked.
2. The HPAGs introduced very little delay (i.e., delays from LAN-to-LAN through the HPAGs were very nearly half the round-trip ping delays between hosts). Transcontinental transit times were on the order of 40 msec.
3. The HPAGs introduced delays of up to 200 msec when retransmitting checksums simultaneously. This "self-synchronization" problem is simple to fix.
4. The HPAG supported the simulation load of 5249 entities with the software resident in user space. By placing the software functionality in the processor kernel, a factor of 10 performance enhancement is probable. Therefore, the HPAG should scale to the STOW ACTD goal of 50,000 entities.

6.1.3 Application-Level Multicast

1. Increasing the grid spacing above the calculated optimum (5 km for high-update-rate grid and 2.5 km for the low-update-rate grid in ED-1A) results in excess traffic flow.
2. Reducing the grid spacing below the calculated optimum appears to have little effect. The expected savings is masked by an increase in the overhead traffic necessary to manage the larger number of address groups.
3. An optimum grid spacing must be determined through experimentation or modeling prior to the commencement of a major exercise. (Logger files from STOW-E were used to calculate the ED-1A optimum grid spacing.)

6.2 QES, SUBSCRIPTION, FIDELITY, AND MULTICASTING INTERACTIONS

Due to the close relationship among the functions of the ACTs performed by agents and the AH, they are grouped together and discussed as follows:

1. A prototype next-generation DIS 3.X protocol was enacted to take advantage of the technologies.
2. A significant portion of the multicast packets on a LAN are irrelevant to any particular simulation. Multicast-based relevance filtering reduces the number of packets that must be processed by simulation applications, making more resources available for simulation processing.
3. Irrelevant multicast packets not addressed to the receiver are generally rejected in the workstation's kernel. This consumes valuable processing resources because interrupt and protocol code must be executed for every packet. The SGI implementation of multicasting suffers significant overhead problems when large numbers of multicast subscriptions are required due to its address search algorithm. Streamlining the search algorithm will reduce packet reception overhead. Better still, hardware filtering of multicast packets would result in a more dramatic reduction in kernel processing.
4. Most packets on the LANs using the new DIS 3.X protocol appeared to be entity-state PDUs, as expected.
5. Packets related to QES constituted a significant portion of the total traffic, as expected. The total traffic levels were reduced, however, by the inclusion of QES. Effectively, control packets related to maintaining a consistent state were substituted for a larger number of state updates. This is due to the fact that on average, roughly two-thirds of the entities in the scenarios examined are quiescent.
6. The QO DP created negligible amounts of traffic.
7. The CP implementation was robust and did not exhibit instabilities or breakdown, even in the face of transient high loss rates, episodes of network connectivity loss, and heavily overloaded simulations.
8. Most CP packets were CP update PDUs. This source of overhead can be reduced significantly by bundling CP update PDUs with entity-state PDUs. This is easy to do since they are generated at the same time and sent to the same destination multicast group.
9. MIB values for entity counts agreed well with values obtained by analyzing the logger files.

10. The Agent Host component was robust and useful. It ran reliably and served as a collection point for exercise statistics. The reliable protocol it employed (CP) for communication with ModSAFs functioned well despite severe overloads.
11. Traffic due to the subscription agent approach is relatively insignificant compared to simulation-related traffic (1-3%).
12. The fidelity/uncertainty technique employed produces a noticeable reduction in traffic (6% to 14% for all but the most heavily loaded test events).

6.3 OVERLOAD MANAGEMENT

Since only the LL aspect of the OM algorithm was implemented, and since LL was not invoked during the various demonstrations, no meaningful findings are available from the ED-1A data. Pre-demonstration tests showed LL to be effective in keeping the traffic load below an established threshold.

6.4 APPLICATION TRANSLATOR

The AT worked well to integrate or bridge a DIS 2.X LAN of simulators into the RITN DIS 3.X network. The QO protocol, as implemented in the AT, was highly successful in reducing packet traffic generated in support of quiescent objects (52.8%). Largely due to the effectiveness of QO, legacy simulators may effectively engage in DIS 3.X exercises without adversely affecting the performance of other ACT components or DIS 3.X simulators. A maximum of 796 entities were observed during post-ED-1A testing, with 44 of those residing on the 2.X LAN.

6.5 TIME SYNCHRONIZATION

Accurate time synchronization was achievable for HPAGs and simulators with relatively little difficulty, but care was required to implement it properly. Additional findings were as follows:

1. Time synchronization procedures need to be automated to reduce human error.
2. A method to ensure that hosts are properly synchronized is needed.
3. All HPAGs remained well-synchronized throughout the exercise.
4. The NRaD WAN data logger had significant time-keeping errors. The cause was hypothesized to be a bug in the SGI operating system.

6.6 QUALITY OF SERVICE

The traffic flows between sites varied over time and were heavily dependent on the scenario. This implies that either over-provisioned static reservations or dynamic reservations will be required in the future. While only 1-minute averages were examined, it was found that generally there were no wild variations from minute to minute. This is encouraging; it indicates that making ATM reservations based on traffic needs may be possible in the future.

6.7 NETWORK IMPLEMENTATION

The network and system surpassed the 5000-entity goal during the largest test event, with 5249 entities over 7 sites and 62 workstations. This is one of the highest active combat entity engagements ever performed with distributed simulation.

Though the WAN was over-provisioned (5 Mbps used, 45 Mbps available on the AAI, 2.4 Gbps available on the ATDnet) and LAN-to-LAN latency was low (less than 70 msec coast to coast), the testing revealed a number of limitations when some components were stressed with very large numbers of multicast groups and heavy traffic loads. The hosts and routers evaluated all exhibited performance degradation of one type or another. These degradations are summarized for each of the components in the following subsections.

6.7.1 Silicon Graphics Workstations

The SGI workstation displayed a sensitivity to ethernet congestion. When an SGI workstation attempted to load the ethernet while competing workstations were injecting high-traffic loads on the same ethernet segment, the SGI workstation would allow the other workstations to "capture" the LAN. With multiple SGI workstations on the LAN, very high ethernet utilization numbers were not attainable with the DIS traffic.

The SGI workstation exhibited anomalous behavior when more than 31 simultaneously active multicast groups were joined using FDDI. The receiving SGI workstation was not able to receive traffic on more than 31 groups at one time. The team observed a deterministic pattern of group numbers that will receive data. There were also some intermittent tests where all data could be received on as many as 1000 groups simultaneously. This issue requires further investigation and discussion with SGI engineers.

6.7.2 Sun Microsystems Workstations

The Sun SPARC workstations suffered limitations when using the FDDI network interface for multicast. Regardless of the packet rate or the number of multicast groups joined, the Sun workstation tended to lose approximately 25% of multicast packets received on the FDDI interface. The Sun was also limited in receiving traffic only on up to 14 groups joined on the FDDI interface at one time.

6.7.3 Cisco 7000 Routers

The Cisco 7000 operating with the Internet Operating System (IOS) 10.2(5.5) (the version that was available prior to ED-1A) was severely limited in its multicast forwarding capabilities. The best that it could do was forward approximately 800 pps with one multicast group, with further degraded performance for larger numbers of active groups.

The IOS 11.0(3.3) release offered greatly improved multicast forwarding performance. However, with large numbers of groups, the Cisco 7000 could not forward significant traffic levels without packet loss between two ethernet LANs. A packet loss rate of approximately 2.5% was experienced when operating at sufficiently high packet rates with large numbers of groups.

6.7.4 Bay Networks Routers

The Bay Networks 72000 router was evaluated running Version 8.11/fix2 of its operating system. Up to 100 groups, the router could forward, without any packet loss, an aggregate load of 7000 pps (test setup limit). Up to 1000 groups, it could forward 6000 pps without loss. When pressed beyond this limit, less than 1% packet loss was experienced up to ethernet/test tool loading limitation of 7000 pps.

The Bay Networks router did exhibit some anomalous behavior in joining large numbers of multicast groups. Sometimes, appreciable delays (up to several minutes) were experienced between the time a host joined a group and when the router began forwarding packets for those groups. This occurred more often when very large numbers of multicast groups were joined at the same time. Bay Networks engineers have reproduced this condition in their lab and are addressing the problem.

6.7.5 LAN

LAN loading consistently reached 3 Mbps, effectively saturating the 10 Mbps ethernet (above this level, the rate of collisions severely degrades the LAN's performance). In the future, other LAN technologies will need to be investigated unless careful attention is paid to the distribution of forces in the virtual battlespace.

6.7.6 Data Collection via SNMP

The ACTS SNMP interface performed satisfactorily during ED-1A. From a centralized location, performance statistics were successfully collected from all the ACT components at all seven sites. ACT components were also controlled via SNMP. Some problems were experienced during data collection and analysis. ACT SNMP subagent processors failed to respond when processors were overloaded. Network overloads caused gaps in data collection due to the use of the UDP/IP protocol. Finally, post processing of the collected data were still required to properly analyze it, and discrepancies in the data packet time stamps (due to improperly synchronized hosts) reduced the data's usefulness.

7. RECOMMENDATIONS

The discussions below, again by functionality, describe the test team's recommendations for additional research, further development, and required provisioning to meet the goals of STOW 97.

7.1 WAN MULTICASTING

WAN multicasting was very successful in ED-1A. The scheme implemented to deliver simulator traffic only where it was needed worked and resulted in lower traffic rates at many points in the system. Work is needed, however, to improve several areas of multicast performance.

7.1.1 Wide-Area Multicast Delivery

During ED-1A, the HPAG forwarding packet rate limit was reached during simulation peaks, and the average rate level was greater than half the maximum rate. The HPAG performance during ED-1A was adequate for the amount of traffic it was required to support. For STOW 97, the goal of 50,000 entities will require a factor of 10 improvement over the current implementation. The approach to improve the performance of the HPAG for its next generation is to run the functions in the kernel space as opposed to the user space. Finally, the technology path to improve the HPAG may not scale as well as routers or smart switches, which implement a bi-level IP/ATM multicast group aggregation from high-density local, as across high-speed ATM backbones. The difficulty occurs when computing the mapping table, given scarce VCs (due to FASTLANE, etc.) and slowness in changing VCs for reservations. Furthermore, to take advantage of ATM point-to-multipoint service, the bi-level multicasting should be implemented where the WAN multicast service is ATM, rather than IP over ATM. In the ED-1A configuration, the routers replicated packets and sent them over multiple point-to-point ATM VCs. As a result, packets were sometimes sent twice over the same link. This can be avoided by using bi-level multicast with ATM point-to-multipoint service. In this configuration, the packet replication would be performed by the ATM switches at link branch points similar to core-based trees of IP multicast. Before making a commitment to using ATM point-to-multipoint service, however, it should be confirmed that it can be reliably provided.

The data gathering process was vastly improved over previous exercises. Work is needed, however, to improve and develop test tools independent of the application that can characterize multicast services. In addition to tools that characterize multicast service join time, data rate, and number of groups, it would be very helpful to have a continuously running multipoint-to-multipoint delay probe system. Such delay information was not available for ED-1A.

7.2 AGENT HOST FUNCTIONALITY

Improved relevance filtering algorithms need to be implemented to provide more effective filtering and to make better use of the limited number of available multicast groups.

The effectiveness of relevance filtering can be enhanced by allocating entities to simulators in order to minimize communication. This can be achieved by putting entities that must exchange data on the same computer or, barring that, on different computers on the same LAN. If possible, the allocation scheme should be dynamic to consider the scenario's evolution.

Circular, rather than rectangular, regions of interest should be investigated to determine if they increase the efficiency of filtering for sensors with a field-of-view that can sweep out a circular region.

An improved multicast implementation will pay large dividends. Hardware filtering will provide the greatest advantage. The vendor community should be pushed to develop better workstation input/output (I/O) architectures. Workstation network interfaces that can more effectively filter out unwanted multicast packets will significantly increase application performance by making more resources available for simulation processing.

The consistency protocol should be applied to other types of data, i.e., transmitters, mines, dynamic terrain, and weather. The current quiescent entity algorithms should be enhanced to support dead reckoning of hulls and articulated parts besides turrets.

The group assignment algorithm employed was simple; however, improvement in filtering techniques will be achieved with better group assignment algorithms such as dynamic clustering.

The agent concept was successfully evaluated. The AH provided a source for application status information, gathering data on entity counts, multicast group utilization, and other information. Because of the distributed nature of the exercise, the AH was the only component that could determine how many entities were actually present. These features should be extended to provide a richer source of information, e.g., packet counts by kind, simulation health, etc.

Agents should be considered useful architectural components and employed whenever appropriate.

Enhanced fidelity channel approaches are needed. Additional mechanisms should be developed to better control and exploit uncertainty tolerance. This will become a more pressing issue as additional WAVs and rapidly steerable sensors are supported in distributed simulation systems.

It is necessary to hold constant as many independent variables as possible to allow direct comparison of experimentally varied parametric runs with controlled baseline runs. This includes using the same scenario files with the same number of entities distributed over the same application hosts on the LAN while varying a parameter such as multicast grid size. ModSAF, the primary application used during these experiments for unit tasking and entity simulation, is an inherent problem. This is due to internal factors that are part of its entity reasoning logic and external factors (such as its cooperation with other applications on the same PO data base via the load-sharing algorithms). Nonetheless, it is prudent to eliminate as many sources of variation as possible to maximize the learning potential during an experiment of this type. In addition, it is necessary to repeat parametric and baseline runs under strictly reproducible conditions to ensure an accurate assessment of the range of the dependent variable under study. There is nothing so satisfying as to have multiple runs corroborating the results.

7.3 OVERLOAD MANAGEMENT

OM was lost from development due to ambitious development schedules and a WAN with insufficient support services. Linked features (QoS and an SRC implementation in ModSAF) were planned, but never completed. As a result, the sophisticated OM function was reduced to dropping packets to limit traffic loads. The recommendations for the future are that the planned work for ED-1A be completed as soon as possible, and that a means of supporting dynamic QoS requests be built into future WANs chosen for employment in distributed simulation exercises.

7.4 APPLICATION TRANSLATOR

Though no performance limitations were exceeded during ED-1A testing, subsequent tests indicated an I/O bottleneck where packets are received off the interface. Depending on the number of legacy simulators expected in STOW 97, a more streamlined design for the AT may be necessary. It may be necessary to split up the functionality of the AT across several workstations, possibly following an agent/principal approach to take care of many of the housekeeping duties currently performed by the AT, such as Entity Regeneration and Expiration Checking. Offloading these types of processes to other applications could help decrease the delay involved in passing PDUs between the legacy and DIS 3.X LANs.

7.5 TIME SYNCHRONIZATION

Time synchronization is a necessary part of the computing and network infrastructure for simulation. In ED-1A, it enabled the use of absolute time stamps by simulators, making one-way delay measurements and in-order log combining possible.

It is recommended that time synchronization be made easier, and that monitoring tools be designed and employed to determine whether it is being achieved. Further work is also needed to enable a user to determine, without requiring expert assistance, whether their chosen operating system and hardware platform can support accurate synchronization.

Future vendor implementations of in-kernel time-keeping will change the nature of the synchronization. Errors can be introduced by new features; clock synchronization verification will have to be done for every new operating system release and hardware combination.

7.6 QUALITY OF SERVICE

The use of multicasting has made QoS support for simulation more complex. This area will become particularly challenging as ATM WAN services begin to be used directly. It is recommended that the issues of QoS support in a multicast/ATM WAN environment be investigated further. Some research and development issues include:

1. Aggregation of Reservations for Multiple Simulators. With a broadcast scheme, all the reservations for simulators at a site could be aggregated to gain significant benefit from statistical multiplexing assumptions. Given that different simulators send traffic to different sets of places, and that these destinations change over time, this issue is far more complex in the multicast domain. However, a benefit can be gained from statistical multiplexing, flows from multiple simulators could be correlated because they are part of the same distributed system.
2. Variation of Reservations and Needed Bandwidth Over Time. Early indications are that signaling rate limitations will constrain ATM SVC setup times to be roughly one per second and, therefore, it seems likely that there will be a mismatch in time scales. Research should be conducted on the subject of prioritizing which SVC reservation to modify and how to send traffic, given the current set of reservations.
3. Bi-level QoS. It must be determined to what extent aggregate reservations for particular application-level multicast groups can be made, and how aggregated reservations can be made for the wide-area group over which those groups are forwarded. Further, it must be determined how the variation over time in both the use of application groups, and the combination of application groups and the sites to which they are forwarded, may combine to produce a variation in the traffic in each WAN group. That is because indications are that the number of

VCs that can be supported in a near-term ATM environment is limited. The bi-level QoS function will also be faced with making use of the scarce resource of available VCs.

4. RSVP to ATM integration. Systems using IP will almost certainly use RSVP for reservation setup. ATM networks use ATM signaling. A network that carries IP over ATM will require some sort of translation of reservations from the IP domain to the ATM domain. This will be a particularly difficult problem as RSVP allows existing reservations to be modified, and the current ATM standards (UNI 3.1) do not. Although it appears that UNI 4.0 will allow modification of existing reservations on point-to-point VCs, no such support appears present for point-to-multipoint VCs. This issue will have to be pursued. An additional problem is that the current RSVP drafts allow for heterogeneous receiver reservations on a multicast group. With ATM, all endpoints of a point-to-multipoint VC must have the same reservation. With IP/RSVP, traffic that does not have a reservation is sent best-effort. Thus, one could have 10 receivers, and only reserved resources going to three of them. The other seven would receive some of the traffic. It is not clear how to implement this functionality efficiently with ATM.

7.7 NETWORK IMPLEMENTATION

The wide-area ATM network was lightly loaded and over-provisioned for ED-1A. Sustained aggregate loading of 3 to 4 Mbps was noted in ED-1A. A 10-fold increase in exercise size (from the maximum 5000 entities in ED-1A to the 50,000 entities projected for STOW 97) is needed to support STOW 97. A linear extrapolation indicates a bandwidth requirement of 30 to 40 Mbps. The RITN test bed for ED-1A could support a STOW 97-level exercise at 30 to 40 Mbps.

An alternative to a 10-Mbps shared ethernet will be needed to scale to meet STOW 97 requirements. Switched ethernet, FDDI, and ATM should be considered. Also, the implications of multicast with the higher speed LAN technologies should be considered.

Data collection via SNMP can be improved in the future by implementation of the following recommendations:

- Assign a higher priority to the SNMP sub-agent than most of the other system processes.
- Put the SNMP Management Platform on a separate LAN directly connected to the router. This will eliminate some of the collisions and should result in better data collection performance.
- Collect "SysTime" from each host with each set of data collected. Since time is collected at the same instant as the statistics, it will provide more accurate time stamping.
- Modify the ACT SNMP interface to recognize other SNMP data types such as "counter," "gauge, "ipaddress, and "timeticks" to reduce the necessity of post-exercise data processing. For example, most of the data variables for the ACT components in ED-1A were of the SNMP variable type "integers." If they were represented, instead, as SNMP variable type "counter," HP OpenView would automatically generate the rate of change in the counter (which is the most useful analysis for counter data) without any post-collection data processing.

8. ACRONYM LIST

AAI	(ACTS) ATM Internet
ACT	Application Control Techniques
ACTS	Advanced Communication Technology Satellite
AG	Application Gateway
AH	Agent Host
API	Application Program Interface
ARL	Applied Research Lab
AT	Application Translator
ATDnet	Advanced Technology Demonstration Network
ATM	Asynchronous Transfer Mode
BBN	Bolt, Beranek and Newman
BRT	Bandwidth-Demand Reduction Technique
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DC&A	Data Collection & Analysis
DIS	Distributed Interactive Simulation
ED-1A	Engineering Demonstration-1A
ESPDU	Entity State Protocol Data Unit
GPS	Global Positioning System
HPAG	High Performance Application Gateway
IDA	Institute for Defense Analyses
Kbps	Kilobits Per Second
LAN	Local Area Network
Mbps	Megabits per second
MC	Multicasting
MIB	Management Information Base
ModSAF	Modular Semi-Automated Forces
NACK	Negative Acknowledgment
NRaD	Naval Command, Control and Ocean Surveillance Center RDT&E Division

NRL	Naval Research Laboratory
NTP	Network Time Protocol
PDU	Protocol Data Unit
POP	Persistent Object Protocol
pps	Packets Per Second
QO	Quiescent Object
QoS	Quality of Service
RITN	Real-Time Information Transfer and Networking
SGI	Silicon Graphics, Inc.
SNMP	Simple Network Management Protocol
SRC	Source Rate Control
STOW	Synthetic Theater of War
STOW 97	Synthetic Theater of War 97
STOW-E	Synthetic Theater of War-Europe
TBD	To be determined
TEC	U.S. Army Topographic Engineering Center
WAN	Wide Area Network

APPENDIX A: ED-1A DATA/STATISTICS FROM SNMP DATA

This appendix presents various statistics calculated from the ED-1A SNMP data collection in tabular form. The team conducted 27 distributed simulation exercises during the 3 days of ED-1A. The simulation scenarios selected for analysis were chosen to assess maximum loading of the LANs and WANs over extended periods of 15 minutes or more. The statistics in this report are based on the following nine runs:

11_14_95_1.1	11_15_95_10.1	11_15_95_4.1	11_15_95_7.1
11_15_95_8.1	11_15_95_8.2	11_16_95_10.1	11_16_95_7.1
11_16_95_max.1			

For each statistic, its maximum value (max), average value (mean), and standard deviation (sdDev) are computed in bits per second (bps). An 'X' represents unavailable data, due to inability to collect data from that component. Statistics are calculated for four types of network components: routers, ModSAF simulators (to measure LAN traffic), HPAGs, and ATM switches. Each site had a set of number designators for each interface. The pattern of number assignment was not uniform across all sites. Table A-1 lists the ports used and their corresponding assigned number.

Table A-1. Site ports and their assigned numbers.

Sites	IDA	NRL26	NRL34	TEC	UTEXAS	NRaD1	NRaD2
router							
LAN	1	1	1	1	1	8	7
ATM	2	2	2	3	4	1	
HPAG	3	3	3	2	2	10	9
switch							
TxWAN		2.0.121				8.4.121	
RxWAN		9.29.121				0.4.121	
ModSAF		1					
HPAG							
LAN		1					
WAN		2					

Each router has three interfaces: one to the local simulation LAN, one to the HPAG via ethernet, and one to the ATM switch. For example, the interfaces of the University of Texas router were labeled 1, 4, and 2, which correspond to the simulation LAN, the ATM switch, and the HPAG, respectively. NRaD had two separate LAN/HPAG sites connected to one router. Later in this appendix, interfaces are classified as either transmit (Tx) or receive (Rx). Router statistics are available at all sites except NRL26.

The ATM switch ports are labeled "to WAN" to indicate traffic outbound from a site, and "to LAN" to indicate traffic inbound to a site. In the case of NRL26, where the following data were collected, switch port number 9 was connected to the ATM cloud and carried outbound traffic, while switch port number 2 carried traffic from the WAN into the local site.

Data in this appendix concerning ModSAF simulators (LAN traffic) and HPAGs are from NRL26 only.

The first group, tables A-2 through A-10, represents SNMP data collected at the various sites. These tables use the directional naming conventions established in figure A-1.

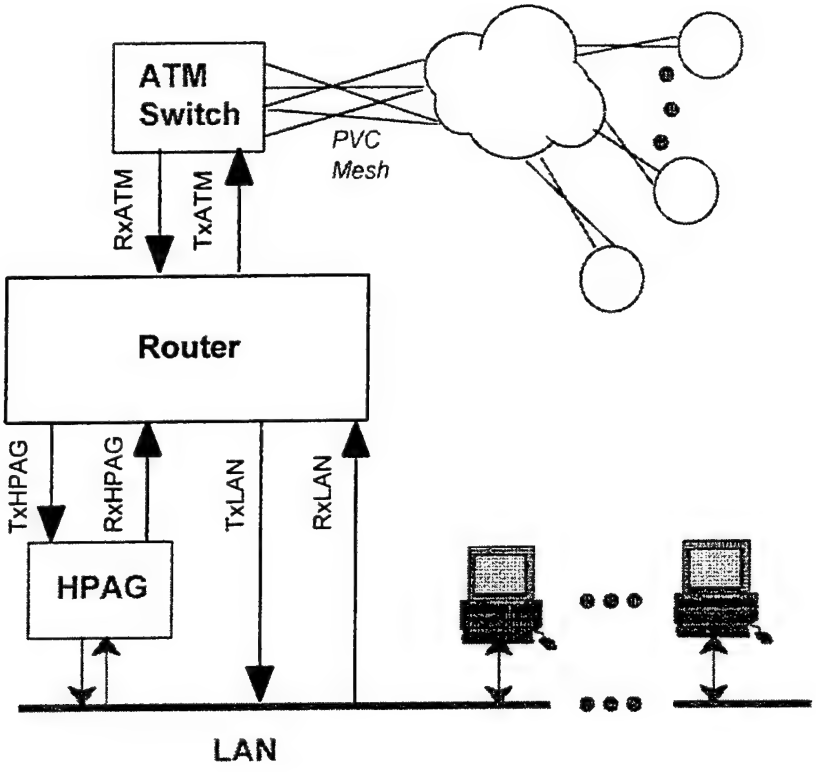


Figure A-1. Router labels.

Table A-2. 11_14_95_1.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRaD1	NRaD2
RxLAN							
max	5.9E+05	X	5.8E+06	3.0E+06	9.2E+05	3.6E+06	1.4E+06
mean	4.0E+05	X	2.2E+06	7.4E+05	3.9E+05	1.9E+06	6.2E+05
sdDev	1.1E+05	X	2.0E+06	6.1E+05	2.7E+05	1.1E+06	4.4E+05
RxATM							
max	1.1E+06	X	2.0E+06	1.0E+06	1.2E+06	7.5E+05	X
mean	4.5E+05	X	6.2E+05	5.0E+05	5.4E+05	3.5E+05	X
sdDev	3.8E+05	X	4.9E+05	3.9E+05	4.3E+05	2.6E+05	X
RxHPAG							
max	2.4E+05	X	1.1E+04	2.6E+05	1.5E+05	3.8E+05	2.4E+05
mean	1.3E+05	X	4.9E+03	1.3E+05	5.9E+04	1.8E+05	7.1E+04
sdDev	9.8E+04	X	2.6E+03	7.9E+04	3.8E+04	1.5E+05	6.9E+04
TxLAN							
max	4.8E+03	X	1.3E+06	4.2E+03	4.3E+03	2.8E+05	6.8E+03
mean	2.4E+03	X	5.6E+04	1.8E+03	2.3E+03	1.8E+04	3.4E+03
sdDev	1.1E+03	X	2.2E+05	9.1E+02	8.9E+02	4.2E+04	1.1E+03
TxATM							
max	6.8E+05	X	2.3E+04	1.8E+06	7.4E+05	2.9E+06	X
mean	3.9E+05	X	1.4E+04	7.8E+05	2.2E+05	1.3E+06	X
sdDev	2.8E+05	X	7.4E+03	6.5E+05	2.0E+05	1.0E+06	X
TxHPAG							
max	3.5E+05	X	1.2E+05	6.2E+05	6.9E+05	8.1E+05	1.0E+06
mean	2.2E+05	X	6.9E+04	4.4E+05	3.6E+05	5.3E+05	6.5E+05
sdDev	8.3E+04	X	5.2E+04	1.7E+05	1.4E+05	1.9E+05	2.4E+05

Table A-3. 11_15_95_10.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRaD1	NRaD2
RxLAN							
max	1.6E+06	X	5.3E+06	1.8E+06	8.2E+05	3.3E+06	2.8E+06
mean	8.1E+05	X	1.7E+06	7.8E+05	4.3E+05	1.6E+06	1.3E+06
sdDev	4.9E+05	X	9.9E+05	6.2E+05	2.9E+05	1.1E+06	7.7E+05
RxATM							
max	1.2E+06	X	1.4E+06	1.4E+06	1.4E+06	1.0E+06	X
mean	5.8E+05	X	7.0E+05	6.6E+05	6.9E+05	4.4E+05	X
sdDev	5.3E+05	X	6.3E+05	5.8E+05	6.2E+05	4.0E+05	X
RxHPAG							
max	3.6E+05	X	1.7E+05	2.7E+05	1.2E+05	5.2E+05	3.4E+05
mean	1.6E+05	X	7.0E+04	9.8E+04	7.3E+04	2.2E+05	1.2E+05
sdDev	1.5E+05	X	6.0E+04	8.6E+04	2.6E+04	1.8E+05	1.1E+05
TxLAN							
max	6.1E+03	X	7.2E+04	3.1E+03	7.4E+03	2.4E+05	7.0E+03
mean	2.9E+03	X	3.7E+04	1.9E+03	3.0E+03	1.8E+04	4.2E+03
sdDev	1.1E+03	X	2.4E+04	6.0E+02	1.2E+03	2.9E+04	1.2E+03
TxATM							
max	1.0E+06	X	1.8E+06	2.3E+06	5.6E+05	3.4E+06	X
mean	4.6E+05	X	9.3E+05	8.8E+05	2.1E+05	1.5E+06	X
sdDev	4.2E+05	X	6.4E+05	8.2E+05	1.9E+05	1.3E+06	X
TxHPAG							
max	5.7E+05	X	9.1E+05	1.2E+06	6.5E+05	1.2E+06	1.2E+06
mean	3.9E+05	X	3.6E+05	6.3E+05	4.5E+05	6.3E+05	8.2E+05
sdDev	1.6E+05	X	1.9E+05	4.5E+05	1.9E+05	4.7E+05	4.7E+05

Table A-4. 11_15_95_4.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEXAS	NRAD1	NRAD2
RxLAN							
max	3.9E+06	X	3.5E+06	2.2E+06	2.0E+06	4.1E+06	4.6E+06
mean	1.1E+06	X	1.7E+06	9.1E+05	7.3E+05	1.7E+06	8.1E+05
sdDev	1.2E+06	X	8.5E+05	6.8E+05	5.9E+05	1.2E+06	1.1E+06
RxATM							
max	1.2E+06	X	1.3E+06	1.2E+06	1.4E+06	1.0E+06	X
mean	3.5E+05	X	4.2E+05	4.3E+05	4.5E+05	2.9E+05	X
sdDev	3.5E+05	X	4.2E+05	3.9E+05	4.2E+05	2.9E+05	X
RxHPAG							
max	4.1E+05	X	1.3E+05	2.3E+05	1.4E+05	4.6E+05	2.7E+05
mean	9.4E+04	X	3.3E+04	7.6E+04	5.8E+04	1.5E+05	9.3E+04
sdDev	1.1E+05	X	3.5E+04	7.4E+04	3.3E+04	1.3E+05	1.0E+05
TxLAN							
max	1.5E+05	X	3.3E+04	1.2E+03	6.4E+03	2.8E+04	7.0E+04
mean	3.9E+03	X	1.6E+04	9.0E+02	1.7E+03	6.8E+03	5.8E+03
sdDev	1.9E+04	X	9.4E+03	1.4E+02	8.1E+02	3.5E+03	1.3E+04
TxATM							
max	1.2E+06	X	9.4E+05	2.1E+06	6.6E+05	3.3E+06	X
mean	2.9E+05	X	5.5E+05	6.0E+05	1.8E+05	1.0E+06	X
sdDev	3.2E+05	X	3.4E+05	6.3E+05	1.8E+05	9.8E+05	X
TxHPAG							
max	1.3E+06	X	1.4E+06	1.3E+06	1.4E+06	1.1E+06	8.9E+05
mean	6.3E+05	X	7.5E+05	7.0E+05	7.5E+05	5.9E+05	5.0E+05
sdDev	2.6E+05	X	3.1E+05	3.0E+05	3.1E+05	2.6E+05	2.2E+05

Table A-5. 11_15_95_7.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEXAS	NRAD1	NRAD2
RxLAN							
max	7.5E+05	X	4.9E+06	5.6E+05	4.5E+05	6.0E+05	5.2E+05
mean	4.0E+05	X	2.1E+06	2.7E+05	2.6E+05	3.5E+05	2.8E+05
sdDev	2.0E+05	X	8.7E+05	1.6E+05	1.4E+05	1.4E+05	1.6E+05
RxATM							
max	5.0E+05	X	6.0E+05	6.1E+05	6.0E+05	5.3E+05	X
mean	2.5E+05	X	3.1E+05	3.3E+05	2.9E+05	2.7E+05	X
sdDev	1.6E+05	X	1.8E+05	1.7E+05	1.7E+05	1.5E+05	X
RxHPAG							
max	2.1E+05	X	1.5E+05	9.0E+04	1.4E+05	9.8E+04	1.2E+05
mean	1.1E+05	X	6.0E+04	2.8E+04	5.3E+04	5.5E+04	3.8E+04
sdDev	5.4E+04	X	4.0E+04	2.6E+04	3.9E+04	2.7E+04	3.6E+04
TxLAN							
max	3.5E+03	X	1.8E+05	2.8E+03	1.6E+05	3.3E+05	2.1E+05
mean	2.5E+03	X	4.0E+04	1.6E+03	6.5E+03	2.2E+04	1.1E+04
sdDev	8.7E+02	X	2.7E+04	6.0E+02	2.5E+04	5.5E+04	3.6E+04
TxATM							
max	6.1E+05	X	2.5E+06	8.4E+05	6.5E+05	9.1E+05	X
mean	3.3E+05	X	1.6E+06	4.3E+05	2.3E+05	4.4E+05	X
sdDev	1.5E+05	X	8.3E+05	2.4E+05	1.9E+05	2.7E+05	X
TxHPAG							
max	3.5E+05	X	2.9E+05	4.7E+05	2.7E+05	5.4E+05	3.5E+05
mean	2.0E+05	X	2.0E+05	2.6E+05	2.1E+05	2.9E+05	2.1E+05
sdDev	9.9E+04	X	6.7E+04	1.2E+05	6.5E+04	1.7E+05	1.1E+05

Table A-6. 11_15_95_8.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRAD1	NRAD2
RxLAN							
max	9.8E+05	X	5.8E+06	6.7E+05	7.1E+05	8.8E+05	6.9E+05
mean	4.5E+05	X	1.5E+06	3.0E+05	3.8E+05	3.7E+05	3.9E+05
sdDev	2.5E+05	X	1.1E+06	2.1E+05	2.3E+05	2.4E+05	2.4E+05
RxATM							
max	6.8E+05	X	7.7E+05	7.9E+05	9.4E+05	6.1E+05	X
mean	2.4E+05	X	3.5E+05	3.0E+05	3.1E+05	3.2E+05	X
sdDev	2.0E+05	X	2.5E+05	2.4E+05	2.5E+05	2.1E+05	X
RxHPAG							
max	1.8E+05	X	1.1E+05	1.2E+05	3.9E+05	9.2E+04	1.1E+05
mean	8.0E+04	X	5.7E+04	6.4E+04	1.5E+05	3.3E+04	5.1E+04
sdDev	6.4E+04	X	4.4E+04	3.9E+04	1.1E+05	2.7E+04	4.0E+04
TxLAN							
max	1.6E+05	X	1.6E+05	1.5E+05	1.6E+05	3.2E+05	3.2E+05
mean	9.1E+03	X	2.6E+04	7.4E+03	9.1E+03	3.0E+04	1.6E+04
sdDev	3.3E+04	X	2.9E+04	3.0E+04	3.3E+04	7.2E+04	6.5E+04
TxATM							
max	5.2E+05	X	1.6E+06	1.2E+06	1.8E+06	8.3E+05	X
mean	2.2E+05	X	8.1E+05	4.8E+05	5.1E+05	3.8E+05	X
sdDev	1.8E+05	X	5.8E+05	3.7E+05	5.5E+05	3.0E+05	X
TxHPAG							
max	5.9E+05	X	6.6E+05	6.8E+05	6.7E+05	6.7E+05	6.9E+05
mean	3.2E+05	X	4.4E+05	3.0E+05	3.8E+05	3.8E+05	3.7E+05
sdDev	1.3E+05	X	1.4E+05	1.9E+05	1.4E+05	2.3E+05	2.2E+05

Table A-7. 11_15_95_8.2—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRAD1	NRAD2
RxLAN							
max	8.5E+05	X	5.0E+06	6.3E+05	6.5E+05	8.0E+05	6.5E+05
mean	4.2E+05	X	1.0E+06	2.9E+05	3.1E+05	3.4E+05	3.2E+05
sdDev	2.2E+05	X	6.9E+05	2.0E+05	2.2E+05	1.8E+05	2.1E+05
RxATM							
max	4.9E+05	X	6.1E+05	6.1E+05	5.7E+05	5.1E+05	X
mean	2.1E+05	X	2.6E+05	2.7E+05	2.3E+05	2.0E+05	X
sdDev	1.7E+05	X	1.9E+05	2.0E+05	1.7E+05	1.6E+05	X
RxHPAG							
max	1.8E+05	X	1.3E+05	7.1E+04	1.9E+05	9.6E+04	1.2E+05
mean	8.4E+04	X	3.5E+04	2.3E+04	6.1E+04	4.5E+04	4.3E+04
sdDev	5.6E+04	X	3.8E+04	2.5E+04	5.3E+04	3.1E+04	3.2E+04
TxLAN							
max	4.8E+03	X	3.2E+04	1.1E+03	6.4E+03	1.8E+04	2.8E+03
mean	1.6E+03	X	1.5E+04	9.3E+02	1.9E+03	6.1E+03	2.1E+03
sdDev	6.7E+02	X	6.0E+03	1.3E+02	9.3E+02	2.7E+03	2.8E+02
TxATM							
max	5.3E+05	X	1.1E+06	8.5E+05	8.9E+05	9.8E+05	X
mean	2.4E+05	X	5.2E+05	3.2E+05	2.6E+05	4.2E+05	X
sdDev	1.6E+05	X	3.1E+05	2.5E+05	2.6E+05	2.9E+05	X
TxHPAG							
max	5.1E+05	X	6.2E+05	6.3E+05	5.9E+05	6.1E+05	6.1E+05
mean	2.9E+05	X	3.4E+05	3.4E+05	3.2E+05	3.2E+05	2.5E+05
sdDev	1.5E+05	X	1.5E+05	1.8E+05	1.3E+05	1.7E+05	2.0E+05

Table A-8. 11_16_95_10.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRAD1	NRAD2
RxLAN							
max	1.5E+06	X	4.9E+06	1.8E+06	7.6E+05	3.1E+06	2.3E+06
mean	8.5E+05	X	1.4E+06	6.0E+05	4.8E+05	1.5E+06	1.1E+06
sdDev	3.2E+05	X	9.1E+05	6.6E+05	2.3E+05	8.0E+05	5.5E+05
RxATM							
max	1.1E+06	X	1.4E+06	1.5E+06	1.4E+06	9.5E+05	X
mean	6.3E+05	X	7.3E+05	8.1E+05	7.6E+05	4.5E+05	X
sdDev	3.8E+05	X	4.5E+05	5.1E+05	4.7E+05	2.8E+05	X
RxHPAG							
max	4.5E+05	X	2.4E+05	9.9E+03	1.6E+05	3.9E+05	4.2E+05
mean	2.1E+05	X	1.2E+05	4.8E+03	6.7E+04	2.1E+05	1.8E+05
sdDev	1.2E+05	X	7.4E+04	2.0E+03	2.4E+04	1.4E+05	1.2E+05
TxLAN							
max	9.7E+03	X	5.8E+04	3.3E+03	1.1E+04	5.1E+04	7.7E+03
mean	3.9E+03	X	2.9E+04	9.2E+02	4.0E+03	1.6E+04	4.3E+03
sdDev	1.6E+03	X	1.5E+04	5.7E+02	1.7E+03	6.7E+03	1.6E+03
TxATM							
max	4.3E+05	X	3.5E+06	2.0E+06	4.7E+05	2.0E+06	X
mean	1.9E+05	X	1.6E+06	1.0E+06	1.5E+05	1.1E+06	X
sdDev	1.2E+05	X	8.9E+05	6.4E+05	1.0E+05	7.1E+05	X
TxHPAG							
max	4.1E+05	X	1.1E+06	2.7E+05	5.9E+05	1.3E+06	1.1E+06
mean	3.4E+05	X	4.6E+05	1.5E+05	4.5E+05	7.8E+05	7.7E+05
sdDev	7.0E+04	X	2.2E+05	1.0E+05	1.1E+05	2.3E+05	2.0E+05

Table A-9. 11_16_95_7.1—Router (bps).

Sites	IDA	NRL26	NRL34	TEC	UTEX	NRAD1	NRAD2
RxLAN							
max	6.7E+05	X	3.9E+06	6.1E+05	5.8E+05	7.9E+05	6.1E+05
mean	3.8E+05	X	5.0E+05	1.9E+05	2.7E+05	3.3E+05	2.9E+05
sdDev	1.6E+05	X	7.1E+05	1.9E+05	2.1E+05	2.4E+05	1.8E+05
RxATM							
max	5.3E+05	X	6.1E+05	6.6E+05	6.3E+05	5.5E+05	X
mean	1.9E+05	X	2.6E+05	2.8E+05	2.4E+05	2.4E+05	X
sdDev	1.8E+05	X	2.1E+05	2.3E+05	2.1E+05	1.9E+05	X
RxHPAG							
max	1.9E+05	X	1.5E+05	7.8E+04	1.5E+05	7.6E+04	1.2E+05
mean	9.9E+04	X	4.1E+04	2.3E+04	7.1E+04	3.1E+04	3.5E+04
sdDev	8.0E+04	X	4.4E+04	2.4E+04	3.2E+04	2.6E+04	4.0E+04
TxLAN							
max	1.5E+05	X	1.5E+05	1.5E+05	1.6E+05	3.2E+05	2.4E+05
mean	6.7E+03	X	1.0E+04	4.0E+03	5.6E+03	1.8E+04	1.2E+04
sdDev	2.2E+04	X	2.0E+04	2.0E+04	2.1E+04	4.8E+04	4.0E+04
TxATM							
max	1.9E+05	X	1.5E+06	1.0E+06	4.4E+05	5.5E+05	X
mean	9.8E+04	X	5.5E+05	4.3E+05	1.4E+05	1.9E+05	X
sdDev	7.7E+04	X	4.6E+05	3.6E+05	1.2E+05	1.8E+05	X
TxHPAG							
max	4.6E+05	X	3.4E+05	5.8E+05	4.4E+05	6.7E+05	4.8E+05
mean	2.5E+05	X	1.5E+05	2.0E+05	3.0E+05	3.4E+05	2.7E+05
sdDev	1.3E+05	X	7.3E+04	1.7E+05	8.9E+04	2.0E+05	1.5E+05

Table A-10. 11_16_95_max.1—Router (bps).

Site	IDA	NRL26	NRL34	TEC	UTEX	NRAD1	NRAD2
RxLAN							
max	1.5E+06	X	5.3E+06	1.4E+06	7.9E+05	5.9E+06	6.4E+06
mean	7.5E+05	X	1.1E+06	5.8E+05	4.3E+05	1.5E+06	1.2E+06
sdDev	4.1E+05	X	8.7E+05	3.0E+05	2.3E+05	8.8E+05	9.9E+05
RxATM							
max	1.1E+06	X	1.2E+06	1.3E+06	1.3E+06	8.3E+05	X
mean	5.4E+05	X	6.0E+05	6.2E+05	6.4E+05	4.1E+05	X
sdDev	4.8E+05	X	5.2E+05	5.4E+05	5.6E+05	3.7E+05	X
RxHPAG							
max	3.0E+05	X	2.4E+05	1.7E+05	1.1E+05	3.5E+05	2.5E+05
mean	1.3E+05	X	8.4E+04	8.9E+04	3.8E+04	1.6E+05	1.2E+05
sdDev	1.3E+05	X	7.5E+04	4.9E+04	3.2E+04	1.4E+05	1.0E+05
TxLAN							
max	6.4E+03	X	1.1E+04	5.0E+03	8.8E+03	3.6E+04	1.1E+04
mean	3.4E+03	X	4.1E+03	1.8E+03	3.5E+03	1.2E+04	4.1E+03
sdDev	1.6E+03	X	1.9E+03	1.0E+03	1.7E+03	5.1E+03	1.9E+03
TxATM							
max	3.0E+05	X	2.1E+06	1.9E+06	3.1E+05	1.6E+06	X
mean	1.3E+05	X	9.9E+05	9.2E+05	1.0E+05	8.1E+05	X
sdDev	1.2E+05	X	8.8E+05	8.2E+05	9.2E+04	7.0E+05	X
TxHPAG							
max	6.0E+05	X	5.1E+05	5.6E+05	5.1E+05	9.9E+05	1.1E+06
mean	3.4E+05	X	3.6E+05	2.5E+05	3.6E+05	7.9E+05	7.6E+05
sdDev	1.5E+05	X	1.4E+05	2.2E+05	1.3E+05	2.8E+05	3.8E+05

The next series of tables reflect LAN traffic from the perspective of a ModSAF host (Conquerant) on the NRL26 site. Tables A-11 through A-18 use the directional naming conventions established in figure A-2 for the ModSAF station (Conquerant) statistics. As noted above, these statistics show LAN traffic levels.

Table A-11. 11_15_95_10.1 Conquerant (bps).

Site	NRL26
RxLAN	
max	6.8E+06
mean	2.0E+06
sdDev	1.6E+06
TxLAN	
max	2.1E+05
mean	8.8E+04
sdDev	6.3E+04

Table A-12. 11_15_95_4.1 Conquerant (bps).

Site	nrl26
RxLAN	
max	4.7E+06
mean	1.7E+06
sdDev	1.6E+06
TxLAN	
max	1.8E+06
mean	1.4E+05
sdDev	3.2E+05

Table A-13. 11_15_95_7.1 Conquerant (bps).

Site	NRL26
RxLAN	
max	1.4E+06
mean	6.1E+05
sdDev	3.7E+05
TxLAN	
max	1.1E+05
mean	2.2E+04
sdDev	3.1E+04

Table A-14. 11_15_95_8.1 Conquerant (bps).

Site	nrl26
RxLAN	
max	4.9E+06
mean	1.2E+06
sdDev	1.1E+06
TxLAN	
max	8.9E+04
mean	1.9E+04
sdDev	2.1E+04

Table A-15. 11_15_95_8.2 conquerant (bps).

Site	NRL26
RxLAN	
max	1.5E+06
mean	8.7E+05
sdDev	5.6E+05
TxLAN	
max	1.0E+05
mean	6.0E+04
sdDev	3.0E+04

Table A-16. 11_16_95_10.1 Conquerant (bps).

Site	NRL26
RxLAN	
max	3.2E+06
mean	1.7E+06
sdDev	9.2E+05
TxLAN	
max	2.4E+05
mean	1.4E+05
sdDev	6.3E+04

Table A-17. 11_16_95_7.1 Conquerant (bps).

Site	NRL26
RxLAN	
max	7.4E+05
mean	3.8E+05
sdDev	2.3E+05
TxLAN	
max	1.3E+05
mean	2.1E+04
sdDev	2.6E+04

Table A-18. 11_16_95_max.1 Conquerant (bps).

Site	NRL26
RxLAN	
max	3.0E+06
mean	1.0E+06
sdDev	1.0E+06
TxLAN	
max	4.3E+05
mean	7.1E+04
sdDev	1.1E+05

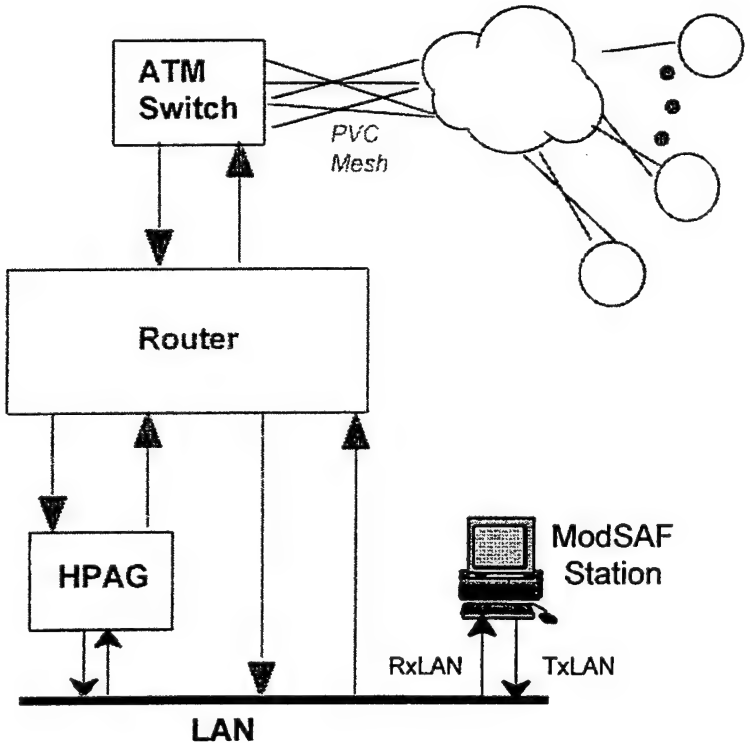


Figure A-2. Workstation labels.

The next series of tables reflect statistics at the High Performance Application Gateway (HPAG). Tables A-19 through A-22 use the directional naming conventions established in figure A-3 for the HPAG at the NRL26 site.

Table A-19. 11_15_95_4.1 HPAG (bps).

Site	NRL26
RxLAN	
max	5.3E+06
mean	1.1E+06
sdDev	1.2E+06
RxWAN	
max	1.3E+06
mean	7.3E+05
sdDev	2.2E+05
TxLAN	
max	1.0E+06
mean	3.8E+05
sdDev	3.3E+05
TxWAN	
max	1.6E+05
mean	6.8E+04
sdDev	5.0E+04

Table A-20. 11_15_95_8.1 HPAG (bps).

Site	NRL26
RxLAN	
max	9.8E+04
mean	6.9E+04
sdDev	3.6E+04
RxWAN	
max	3.9E+02
mean	3.9E+02
sdDev	6.9E+00
TxLAN	
max	X
mean	X
sdDev	X
TxWAN	
max	X
mean	X
sdDev	X

Table A-21. 11_16_95_10.1 HPAG (bps).

Site	NRL26
RxLAN	
max	5.3E+06
mean	1.2E+06
sdDev	9.5E+05
RxWAN	
max	6.6E+05
mean	4.2E+05
sdDev	1.4E+05
TxLAN	
max	5.8E+05
mean	2.8E+05
sdDev	1.7E+05
TxWAN	
max	2.1E+05
mean	1.1E+05
sdDev	3.5E+04

Table A-22. 11_16_95_7.1 HPAG (bps).

Site	NRL26
RxLAN	
max	5.1E+05
mean	3.4E+05
sdDev	8.9E+04
RxWAN	
max	3.5E+05
mean	2.8E+05
sdDev	3.6E+04
TxLAN	
max	5.3E+05
mean	2.2E+05
sdDev	8.2E+04
TxWAN	
max	1.0E+05
mean	2.8E+04
sdDev	2.4E+04

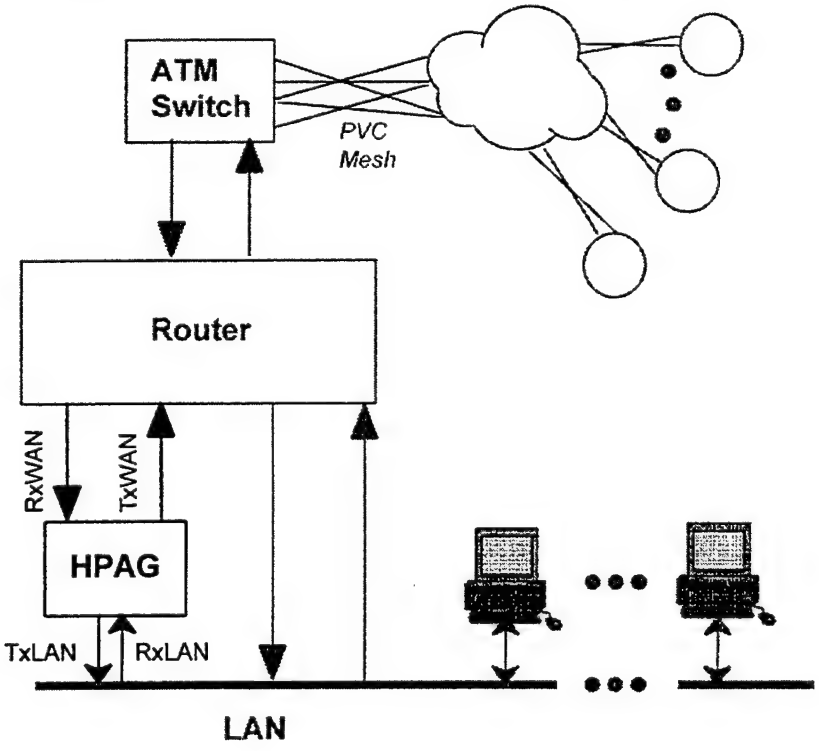


Figure A-3. HPAG labels.

The final series of tables reflect statistics at the ATM switches at NRL26 and NRaD. Tables 23 through A-28 use the directional naming conventions established in figure A-4 for the ATM switch statistics.

Table A-23. 11_15_95_8.1 switch (bps).

Sites	NRL26	NRaD
TxWAN		
max	2.6E+05	8.3E+05
mean	1.3E+05	4.4E+05
sdDev	7.8E+02	8.2E+02
RxWAN		
max	8.3E+05	2.6E+05
mean	4.4E+05	1.3E+05
sdDev	8.2E+02	7.8E+02

Table A-24. 11_15_95_10.1 switch (bps).

Sites	NRL26	NRaD
TxWAN		
max	5.2E+05	2.3E+05
mean	5.3E+04	1.1E+05
sdDev	1.0E+03	9.6E+02
RxWAN		
max	2.2E+05	5.2E+05
mean	1.1E+05	5.3E+04
sdDev	9.6E+02	1.0E+03

Table A-25. 11_15_95_8.2 switch (bps).

Sites	nrl26	nrad
TxWAN		
max	3.6E+04	1.9E+05
mean	2.6E+04	1.2E+05
sdDev	1.2E+03	1.2E+03
RxWAN		
max	1.9E+05	3.6E+04
mean	1.2E+05	2.6E+04
sdDev	1.2E+03	1.2E+03

Table A-26. 11_16_95_10.1 switch (bps).

Sites	nrl26	nrad
TxWAN		
max	2.7E+05	8.1E+05
mean	1.2E+05	4.7E+05
sdDev	8.8E+02	8.8E+02
RxWAN		
max	8.2E+05	2.7E+05
mean	4.7E+05	1.2E+05
sdDev	8.8E+02	8.8E+02

Table A-27. 11_16_95_7.1 switch (bps).

Sites	NRL26	NRaD
TxWAN		
max	7.0E+05	2.3E+05
mean	8.0E+04	8.4E+04
sdDev	9.9E+02	8.4E+02
RxWAN		
max	2.3E+05	6.8E+05
mean	8.4E+04	8.0E+04
sdDev	8.4E+02	9.9E+02

Table A-28. 11_16_95_max.1 switch (bps).

Sites	NRL26	NRaD
TxWAN		
max	2.8E+05	6.8E+05
mean	1.3E+05	3.5E+05
sdDev	7.4E+02	7.5E+02
RxWAN		
max	6.8E+05	2.8E+05
mean	3.4E+05	1.3E+05
sdDev	7.4E+02	7.4E+02

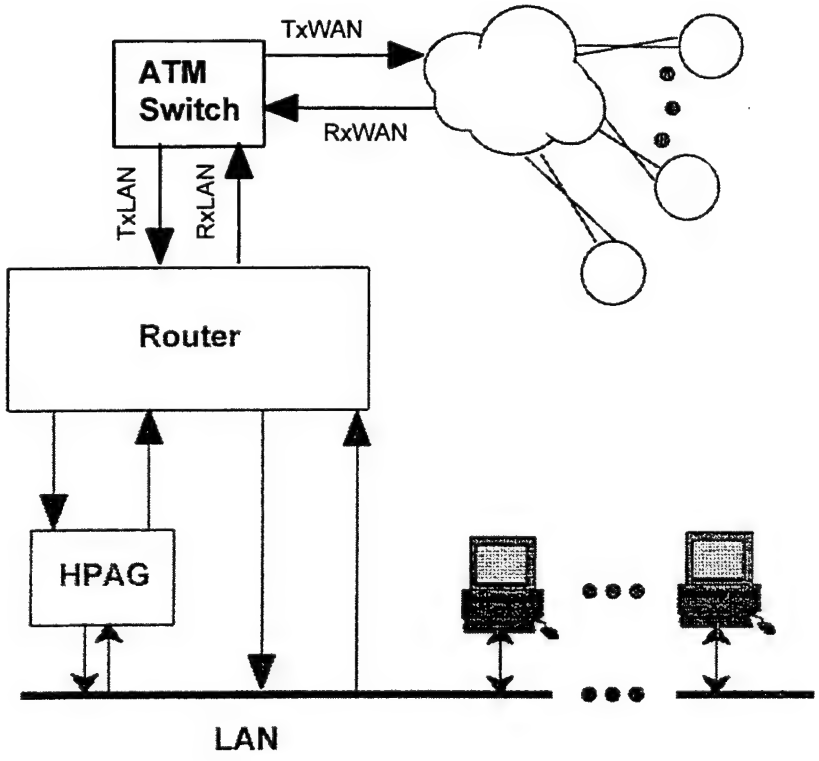


Figure A-4. ATM switch labels.

Appendix B
Analysis of WAN Multicasting Data

Gregory D. Troxel <gdt@bbn.com>
Michael A. Patton <map@bbn.com> Timothy J. Shepard <shep@bbn.com>

31 January 1995

1 Introduction

In this appendix we analyze data from ED-1A to understand how multicast was used in the wide-area network. In particular, we examine data from the HPAG to understand how much simulation traffic was forwarded from the various sites to the WAN, across the WAN, and from incoming WAN links onto the various site LANs.

This appendix contains graphs derived from data collected from the HPAGs during ED-1A, explanations of how those graphs were derived, and analysis of those graphs.

All of the graphs appear after the text. There is one page per exercise; each page contains multiple views of that exercise. The top four plots are the same on all pages. The bottom two plots vary by event. Each graph or set of related graphs is described in a separate section.

Throughout the discussion, we compare what happened in ED-1A to the "broadcast case". The "broadcast" scenario is one in which all data is forwarded to all sites, similar to the scheme used in the DSI. For this case, we presume that every simulation packet generated on any site LAN is sent to all other sites and appears on every remote LAN. When we consider the benefits of multicast, this is the reference.

2 Source data

The data used to compute the graphs come from sampling MIB variables from the HPAG. As in all standard MIBs, the variables counted packets as they went by. Packet rates were computed by taking the difference between two consecutive samples and dividing by the time between those samples.

The variables were polled once a minute, but, as in most SNMP sampling, some samples are missing. In those cases where only a few samples are missing, the rate can be computed over a longer interval between the existing samples. If the missing data was for more than five minutes, no rate is computed and instead a gap appears in the graph to represent the missing data. Thus the rates used in computations are one to five minute averages.

A further complication is that we wish to compare packet rates at multiple HPAGs; the sampling times at multiple HPAGs were in general not identical. To deal with this problem, we interpolated counter values for times that were not available. For example, when dividing the amount of data forwarded onto a LAN by one HPAG by the total data received on the LANs of all other HPAGs,

we used the times for which we had data on the HPAG under scrutiny and interpolated values for the transmitting HPAGs. While imperfect, this approach allowed us to obtain meaningful values in many cases.

However, in a significant number of exercises, there was a sufficiently large amount of missing data that some graphs simply could not be produced. In particular, data was missing on several occasions from one of the NRaD HPAGs.

The HPAG counted packets at several points:

Received from LAN The number of multicast packets received from the LAN by the HPAG. This includes every multicast packet sent by a simulator or agent, and does not include any multicast packets transmitted by the HPAG itself

Transmitted to WAN The number of multicast packets transmitted to the WAN by the HPAG. This number is necessarily smaller than the "received from LAN" number, as each packet is either discarded or forwarded once.

For some exercises, this data is also available as individual counts per WAN group.

Received from WAN The number of multicast packets received from the WAN. If there is no duplication of data in the WAN, this should be less than the sum of the packets transmitted to the WAN by the other sites.

Transmitted to LAN The number of multicast packets transmitted to the LAN by the HPAG. This number is necessarily smaller than the "received from WAN" number, as each packet is either discarded or forwarded once.

3 Types of graphs

We produced several types of graphs from the raw data. These range from simple rate calculations — the rate of change of a counter — to ratios comparing how much data was received compared to how much *would have been* received in a broadcast environment.

Together, the graphs show us how much data was received on the LAN, and how much was forwarded to the WAN. They then show how much was received from the WAN and delivered to the LAN. By combining data from multiple sites, we can compute the effect of multicast in reducing traffic to destination LANs as well as in reducing traffic to the tail circuits and site routers of destinations.

There is one page per exercise for which we have data. Each contains six graphs. The first four graphs are the same for each exercise. The last two either show data derived from the per WAN group counters or additional data derived from the basic four counters.

All of these graphs show rates, and are therefore differences over time of the basic counters. Most show values over one minute intervals, which was the basic data collection interval.

3.1 Basic Data and Forwarding Rates

The upper left graph for each exercise shows the LAN input rates for the various sites. This is, in a sense, the input or driving function to the network.

The upper right graph shows the "WAN-bound forwarding ratio". This is the number of packets sent to the WAN divided by the number received from the LAN. This ratio would be exactly one if all received packets were forwarded to the WAN, and zero if none were. It cannot be greater than one, and plotted values that are greater than one are due to time differences in sampling the different counters. Note, however, that it is usually very close to one, and that values greater than one are immediately preceded or followed by values a corresponding amount less than one.

The middle left graph shows the "LAN-bound forwarding ratio". This is the number of packets sent to the LAN divided by the number received from the WAN. Like the previous ratio, it would be one if all packets were forwarded and zero if none were; it cannot be greater than one.

For exercises 1, 2, 3, 4, 6, 8, 9, and 11, the lower left graph shows the WAN input rate. For these exercises the lower right shows tail circuit reduction rates, described in Section 3.2.

For the remainder of the exercises, the bottom two graphs show data rates from one site to all other sites; these are described in Section 3.3.

3.2 Overall WAN Multicast Reduction

The middle right graph shows "overall multicast reduction", with zero indicating no data delivered and one indicating an amount of data corresponding to the broadcast case. One can convert a value of 0.20 to a "reduction factor" of 5, but we feel that the view which shows what fraction of the data was delivered compared to what would have been delivered is more helpful in understanding the system.

"Overall multicast reduction" is computed by dividing the amount of data sent onto a LAN during an interval by the sum of the data that was originated on all other LANs. If one labels the time derivatives of the four basic counter values as follows, we can then express multicast reduction precisely.

O packet rate received on LAN (for originate)

F packet rate transmitted to WAN (for forwarded)

R packet rate received on WAN (for received)

D packet rate transmitted to LAN (for delivered)

Then, we can use O_i to refer to the packet rate observed at HPAG i.

Overall multicast reduction for site i is then given as:

$$\frac{D_i}{\sum_{j \neq i} O_j}$$

For some exercises, the lower right graph shows the "tail circuit" reduction. This is similar to the overall reduction, but measures the data flow into a site rather than onto a site's LAN. Tail circuit reduction for site i is then given as:

$$\frac{R_i}{\sum_{j \neq i} O_j}$$

Overall reduction tells us how successful we were in reducing traffic sent to the site LAN. Tail circuit reduction shows us how successful we were in reducing data sent to a site's HPAG.

There is a limit to the amount of overall reduction which can be achieved by the network given multicast group membership; all data sent to a group must be delivered to all sites which are members. Given a degree of overall reduction, an efficient network scheme would achieve exactly the same degree of tail circuit reduction, as data would only be sent to a site when it was needed.

We should point out that there are benefits to multicast beyond what is shown here. Packets addressed to a multicast group that has one member on a LAN will be delivered to that LAN, but may be rejected by the hardware or operating system of a simulator. In the broadcast case they would have reached the application.

3.3 Per-WANgroup Rates

For some exercises, we recorded the number of packets sent to each WAN group. Because we know the membership of the HPAGs in the various WAN groups, we can compute how much of a particular site's output traffic was sent to the various other sites. These graphs appear as the lower left graph for exercises 7, 10, 13 and MAX. They show data rates from the UT HPAG (chosen as a representative example) to other remote sites.

We can see that the data rates going to several remote sites are very close to each other and larger than the others. In general, closer examination of the base data indicates that much of the data was sent to more than one but less than all of the remote sites, typically with a peak around 2 or 3 remote sites.

If one lets F_w be the forwarding rate from a site to a particular WAN group w, then we computed the data rate from that site to site A as:

$$\sum_{w|\text{siteA}\in w} F_w$$

The lower right graph is another view of this same data, showing the fraction of outgoing data sent to each remote site. In other words, we computed

$$\frac{\sum_{w|\text{siteA}\in w} F_w}{\sum_w F_w}$$

4 Per-exercise interpretation and analysis

In this section we discuss the graphs for each exercise. We are most interested in the multicast parameters for the exercise, and do not attempt to draw conclusions regarding the effects of simulator-specific ACT algorithms such as QES.

The graphs are labeled either `WAN multicast` or `no WAN multicast`, referring to whether we planned to use multicast effectively in the WAN. The graphs are also labeled either `LAN multicast` or `no LAN multicast`, referring to whether we planned to use multicast effectively in the LAN.

`WAN multicast` The HPAGs were configured to use 127 WAN groups, which is the "optimal" number for 7 sites ($127 = 2^7 - 1$). Thus, a packet always could be forwarded over a WAN group that included exactly the intended set of receivers.

`no WAN multicast` The HPAGs were configured to use 1 WAN group. In this configuration, a packet could be forwarded to the WAN or dropped. However, if it is forwarded at all, it must go to the single WAN group and thus it is received at all other sites.

`LAN multicast` Many of the exercises used on the order of 1000 application-level multicast groups.

`no LAN multicast` Some exercises used only 1 multicast group. Since every host subscribed to it, this effectively converted the entire network to a "broadcast scenario."

Given these descriptions, there are three interesting combinations which we used for various exercises.

no LAN, no WAN This is equivalent to the broadcast scenario. All data was sent to all sites and appeared on all LANs.

Events 4, 6, and 8 had this combination.

LAN, no WAN This used multicast on the LANs. Data was sent to all sites' HPAGs if it was needed at any other site. Data was only forwarded onto the remote LANs if it was needed. Essentially, this used multicast from the simulators' point of view, but didn't use it from the network link and HPAG processing load points of view.

Events 2 and 9 had this combination.

LAN, WAN This used multicast both on the LANs and in forwarding across the LANs. This situation is multicast used by the simulators as we intended and implemented by the network as we intended. While this situation and the previous seemed the same from the simulators's point of view, this configuration required less network resources.

Events 1, 3, 7, 10, 11, 13 and "max" had this combination.

For all events, the WAN-bound forwarding ratio was close to one; almost all data was needed by at least one other site.

4.1 Event 1

Event 1 was the large scenario and used both LAN and WAN multicast.

LAN input rates ranged from about 30 pps to about 100 pps, with peaks of around 130 pps. WAN input rates were about 500 pps for one NRaD site, and about 200 pps for most other sites.

The LAN-bound forwarding ratio was close to one. This indicates that data was only sent to sites where the HPAG then decided to forward it to the LAN. This indicates that the bilevel multicast routing protocol functioned correctly.

Overall and tail circuit reduction rates were comparable. The two NRaD sites showed very little reduction, receiving 90% of the broadcast case traffic, but all other sites received from 20% to 70% of the broadcast case traffic.

We have no explanation for the rapid drop and recovery of traffic at about 11:35; we have no reason to believe it had anything to do with multicasting or the HPAG.

4.2 Event 2

Event 2 was the large scenario and used LAN multicast but not WAN multicast.

LAN input rates ranged from about 30 pps to about 120 pps, with peaks of around 140 pps. These rates are similar to event 1. WAN input rates were about 400 to 600 pps for all sites over the exercise, and the variation at any one time among sites was roughly 100 pps.

The LAN-bound forwarding ratios varied considerably, ranging for 0.2 for some sites to close to 1 for others. This indicates that data was sent to sites where the HPAG then decided to forward some of it and drop some of it. This is as expected since only one WAN multicast group was used; any data forwarded to the WAN went to all sites.

Overall and tail circuit reduction rates were strikingly different. For overall reduction, the two NRaD sites showed very little reduction, receiving 80% to 95% of the broadcast case traffic, but all other sites received from 20% to 70% of the broadcast case traffic. Tail circuit reduction was not observably different from 1, meaning that all sites received about the same amount of traffic that they would have received in the broadcast case.

Examined together, the overall and tail circuit reduction rates show that while the use of LAN multicast reduced data on the simulation LANs, there was no reduction of the amount of data delivered to the tail circuits.

4.3 Event 3

Event 3 was the large scenario and used both LAN and WAN multicast. Unlike event 1, QES was not in use.

LAN input rates ranged from about 50 pps to about 250 pps. WAN input rates were about 700 pps for one NRaD site, and about 100 to 400 pps for other sites. We are not comfortable attributing the greater traffic loads to the lack of QES because we have no reason to believe that the ways the scenario was played out were equivalent, but the lack of QES does seem to be at least a plausible explanation for the higher loads.

The LAN-bound forwarding ratio was close to one. This indicates that data was only sent to sites where the HPAG then decided to forward it to the LAN. This indicates that the bilevel multicast routing protocol functioned correctly.

Overall and tail circuit reduction rates were comparable. The two NRaD sites showed very little reduction, receiving 80% to 90% of the broadcast case traffic, but all other sites received from 15% to 50% of the broadcast case traffic. This corresponds to a reduction ratio between 4 and 6 for non-NRaD sites.

4.4 Event 4

Event 4 was the large scenario and used neither LAN nor WAN multicast. Thus, this event is in fact the "broadcast case" we have used as a reference.

LAN input rates ranged from about 20 pps to about 150 pps. WAN input rates were about 250 to 600 pps, and were within 100 pps of each other at any particular time.

The LAN-bound forwarding ratio was close to one. This is as expected because every site subscribed to the single multicast group in use.

Overall and tail circuit reduction rates were both essentially 1, meaning that each site received 100% of the traffic it would have received in the "broadcast case". This result validates our comparison approach. This is perhaps more important than it might seem because in computing reduction rates we are dividing the rate of data delivered to one LAN by the data originated at other WANs. This implies that

- Our mechanisms to compare rates over different (but nearly the same) time intervals is sound.

- Our basic assumption that almost all data sent across the WAN was delivered is sound. If this were not true then less data would have been delivered to LANs than was originated at remote sites.

During this event the HPAG handled a 1-minute peak of about 700 pps. The fact that computed reduction ratios were not significantly different from one indicates that there were no gross failures to deliver data. This is consistent with lab results indicating that the HPAG could forward 1100 pps.

4.5 Event 6

Event 6 was the small scenario with no multicast and no ACT algorithms.

Data rates were lower than event 4, but overall and tail circuit reduction rates were 1, indicating no reduction. This is as expected since this was also a "broadcast case" exercise.

4.6 Event 7

Event 7 was the small scenario and used both LAN and WAN multicast.

LAN input rates during the beginning ranged from about 20 pps to about 90 pps. The LAN-bound forwarding ratio was close to one. This indicates that data was only sent to sites where the HPAG then decided to forward it to the LAN. This indicates that the bilevel multicast routing protocol functioned correctly.

Reduction rates could not be computed due to missing data.

Data rates are shown from UT to all remote sites, along with the fraction of the total UT output sent to each site. These graph shows that UT sent almost all of its data to three sites. One of the NRaD sites received about half the data, but the same fluctuations apparent in the total output are apparent in this site's data. One site received very little data and one site received little data at the beginning but then received most of the data towards the end.

4.7 Event 8

Event 8 was the small scenario with no multicast. WAN input rates were similar at all sites. The LAN-bound forwarding was 1. Reduction rates could not be computed due to missing data.

The data from this exercise is as expected; with no multicast each site HPAG appeared to receive all the data and forward it to their LANs.

4.8 Event 9

Event 9 was the small scenario with LAN multicast but no WAN multicast. WAN input rates were similar at all sites. The LAN-bound forwarding ratio ranged from 0.2 to close to 1. Reduction rates could not be computed due to missing data.

The data from this exercise is as expected; with no WAN multicast each site HPAG appeared to receive all the data, but due to LAN multicast not all of the data was forwarded to their attached LANs.

4.9 Event 10

Event 10 was the large scenario with both LAN and WAN multicast. An alternate LAN grid spacing was used.

LAN input rates ranged from 30 pps to about 170 pps. The LAN-bound forwarding ratio was 1.

Examining the data rates from UT to other sites, we see that some sites needed most of the data, some needed only a small fraction (0.2) of it, and one needed little in the beginning but most in the end.

4.10 Event 11

Event 11 was the large scenario with both LAN and WAN multicast.

LAN input rates ranged from 40 pps to 250 pps. WAN input rates ranged from 250 pps to 600 pps.

The overall and tail circuit reduction rates were essentially the same. They were close to 1 for the two NRaD sites, and ranged from 30% to 60% of the "broadcast case" traffic.

4.11 Event 13

Event 13 was the small scenario with synthetic environment and both LAN and WAN multicast.

LAN input rates ranged from 10 pps to 50 pps. The LAN-bound forwarding ratio was essentially 1.

No reduction rates could be computed due to missing data.

Data rates from UT to all sites showed similar patterns to previous exercises, with some sites getting most data, some getting some of it, and some getting a greater fraction as the event progressed.

4.12 Event MAX

The event "MAX" was the large scenario with both LAN and WAN multicast. The LAN input rates ranged from 30 pps at one site to about 150 pps at the busiest, with a broad distribution across sites. The LAN-bound forwarding ratio was 1. Overall multicast reduction could not be computed due to missing data.

Data rates from UT to all sites showed that some sites needed most of the data, while some needed only abut 25%. This indicates that the multicast scheme was working.

5 Conclusions

5.1 Data Collection

The design of future data collection systems should ensure

- Collection of desired data with high assurance. We believe that this precludes the use of SNMP as the primary mechanism for gathering data for post-exercise analysis.
- Accurate timestamping of data when it is sampled. The timestamps in the data used for this report have a one second granularity, and were taken at the receiving management station, not at the source.

- Collection of data at precisely the desired times. We believe that many of the anomalies in graphs in this report are due to comparison of rates derived from samples taken at slightly different times.

5.2 HPAG performance

While the data in the graphs that follow cannot be directly used to determine the forwarding performance of the HPAG, lab tests indicate a throughput of roughly 1100 pps. Examining many of the graphs, we see that most of the HPAGs were only called on to forward on the order of 800 pps (sum of both directions) much of the time. HPAG performance seemed just barely adequate for ED-1A. Thus, it is clear that roughly a factor of ten improvement will be needed to support a 50,000 entity exercise. However, because we expect that a larger exercise will have a similar local density of entities and thus more geographic separation, it seems likely that it will show substantially better traffic separation due to multicast. Thus, the input data rate to a site should increase by less than the factor of ten predicted by simple linear extrapolation.

5.3 Use of Multicast

ED-1A used multicast to reduce

- the traffic on tail circuits and site routers
- the traffic on site LANs
- the traffic received by simulation applications

This report addresses only the first two uses of multicast, and primarily focuses on multicast reduction in the large scenario case.

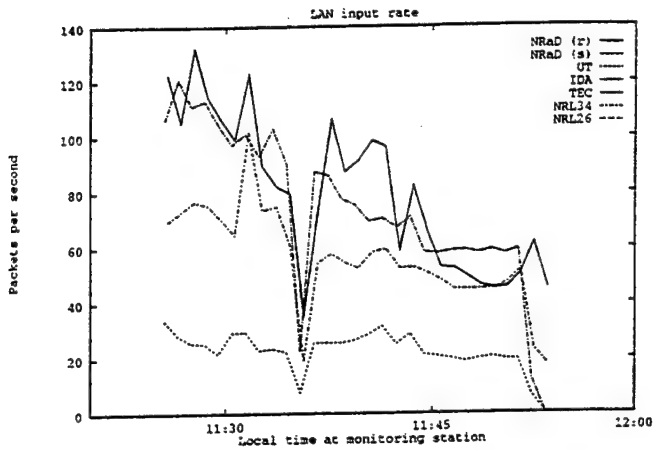
Little reduction was observed for the NRaD sites, which generally received 90% of the traffic they would have received in the broadcast case. We believe that this is due to the central role the blue forces at NRaD played in the scenario and to the fact that the NRaD had far more simulation equipment than almost all other sites. Simulators at NRaD subscribed to most active multicast groups, and thus most data was delivered there.

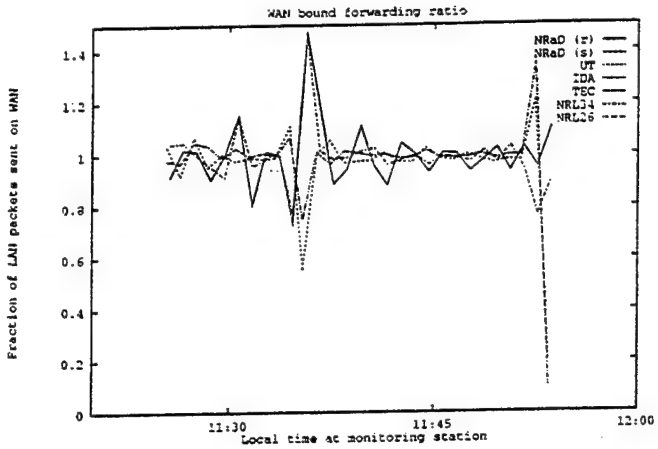
For sites other than NRaD, we observed that sites received an amount of traffic equal to from 15% to 70% of the traffic that would have been received in the broadcast case. This corresponds to a reduction ratio between 1.4 and 6.

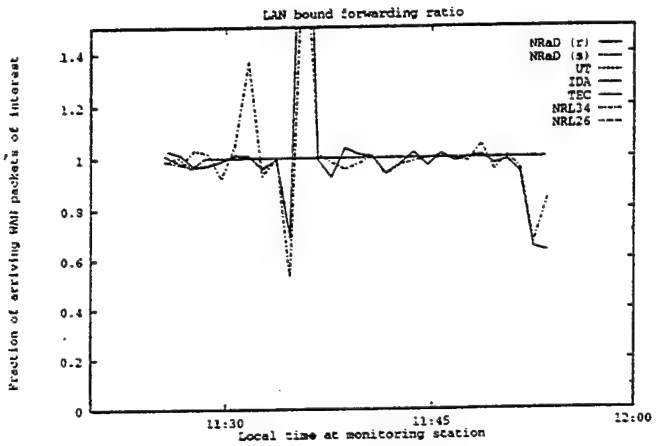
5.4 Recommendations

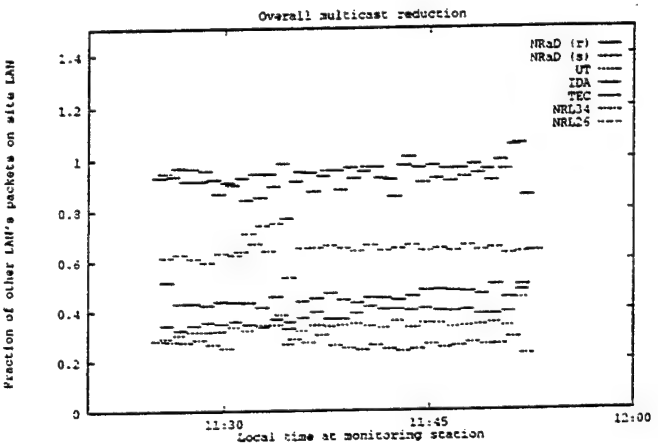
We recommend that careful attention be paid to analysis requirements in the design of future data collection systems.

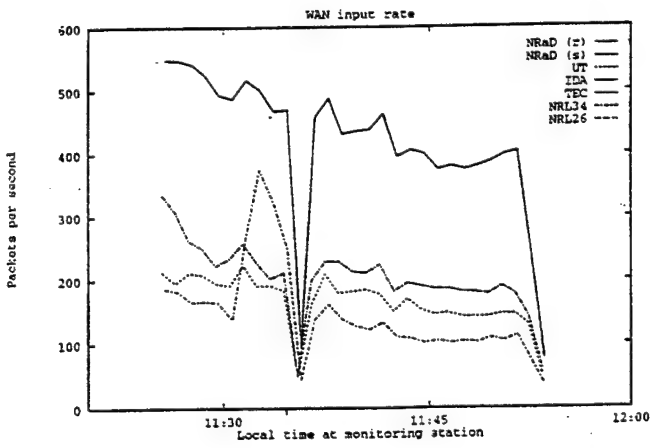
We recommend that the HPAG functionality be reimplemented in a way that will give at least a factor of ten improvement in forwarding performance.

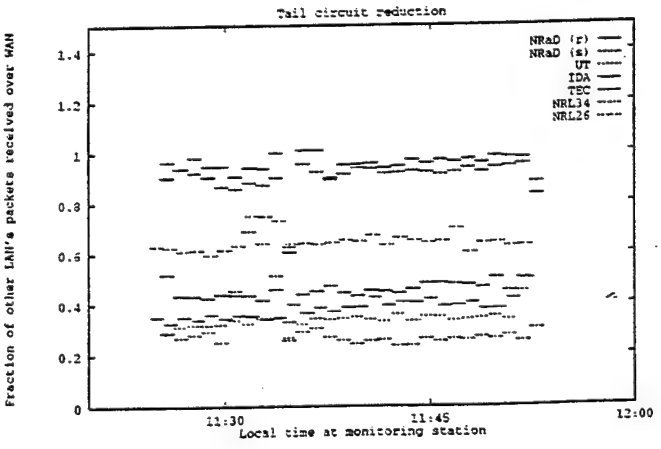






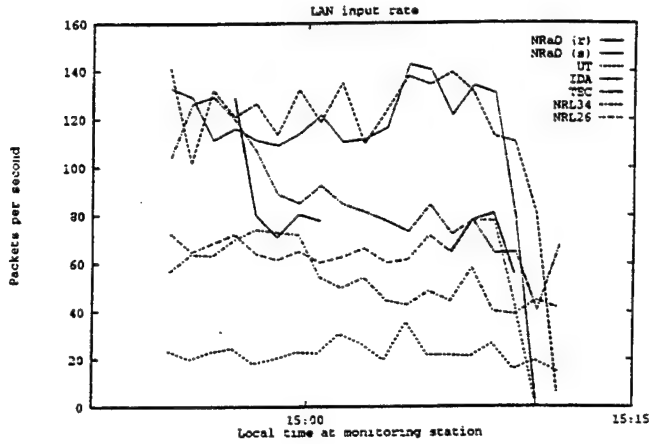


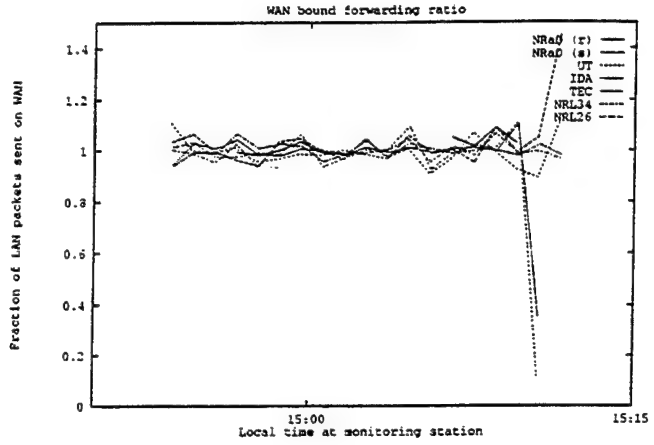


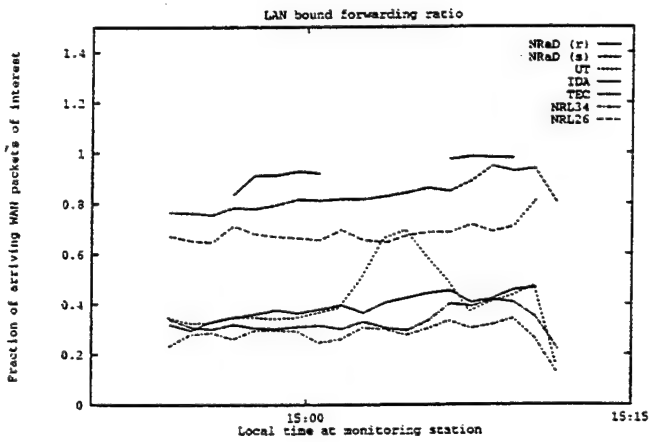


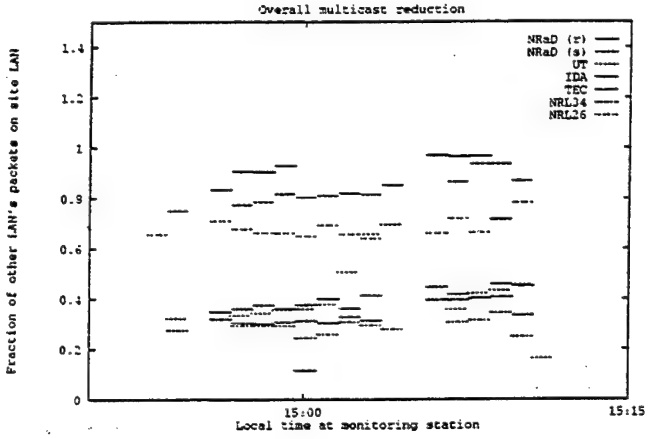
Run on 14-Nov-95 data from 11:20 to 12:00
Large Scenario
All algorithms
WAN multicast, LAN multicast

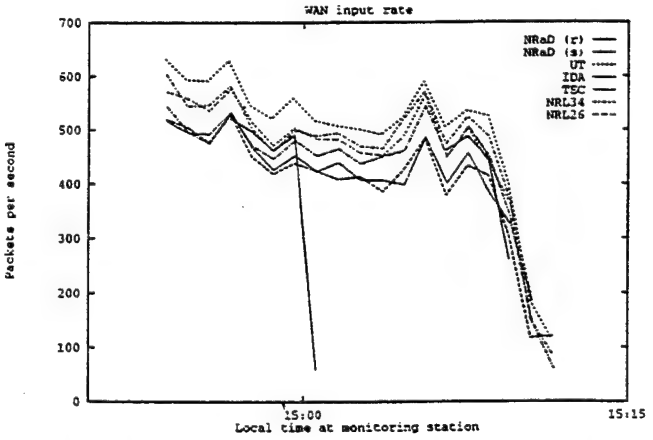
Event 1

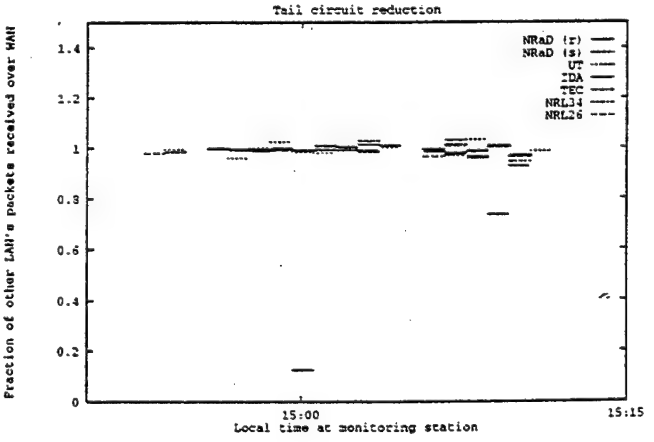






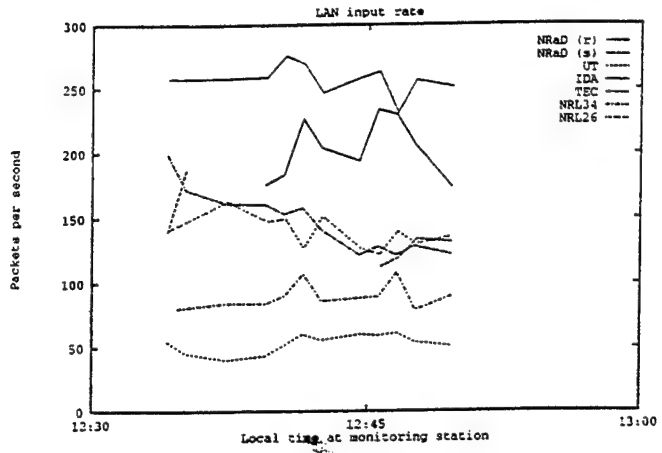


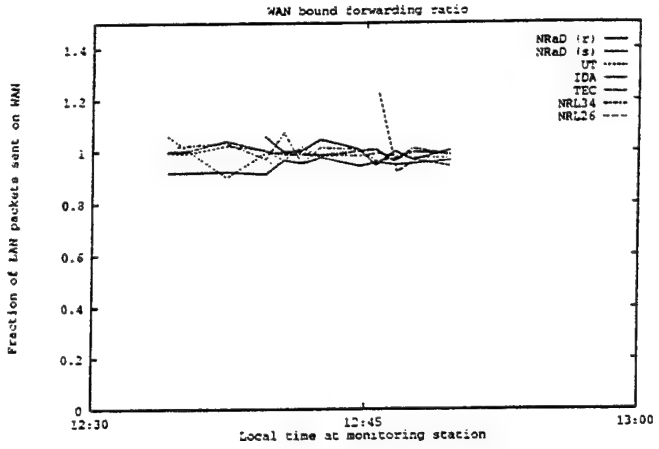


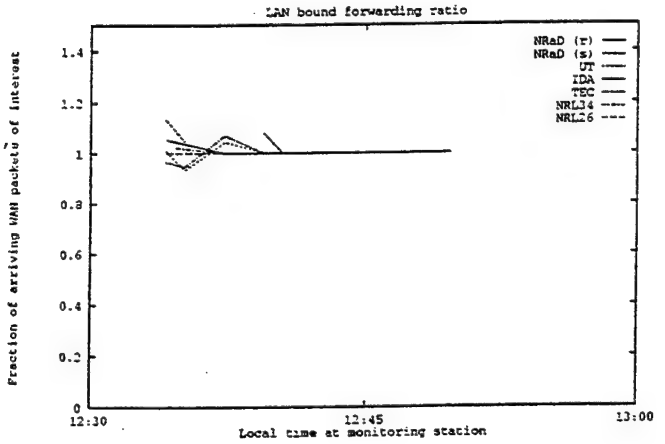


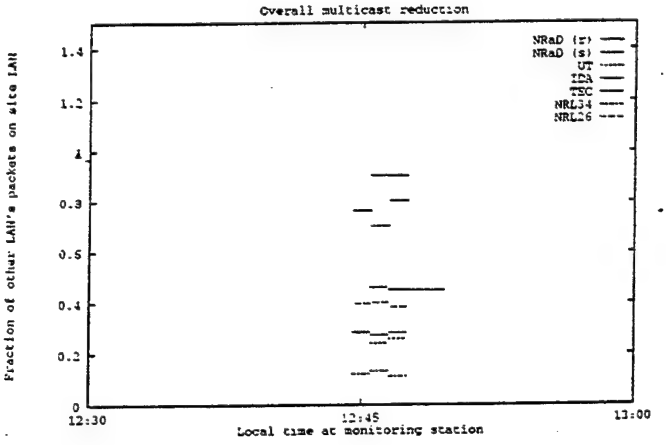
Run on 14-Nov-95 data from 14:50 to 15:15
Large Scenario
QES, LM
no WAN multicast, LAN multicast

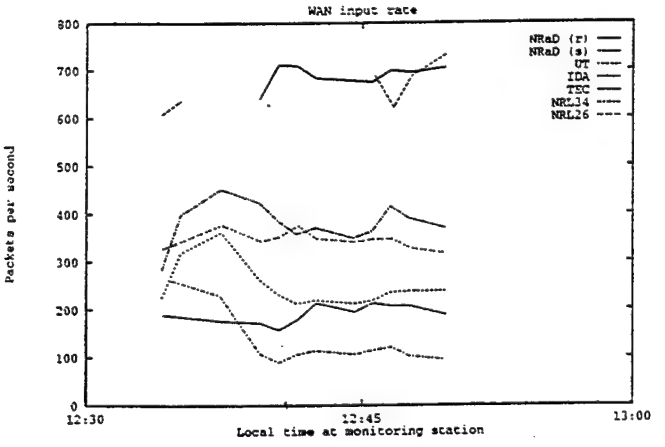
Event 2

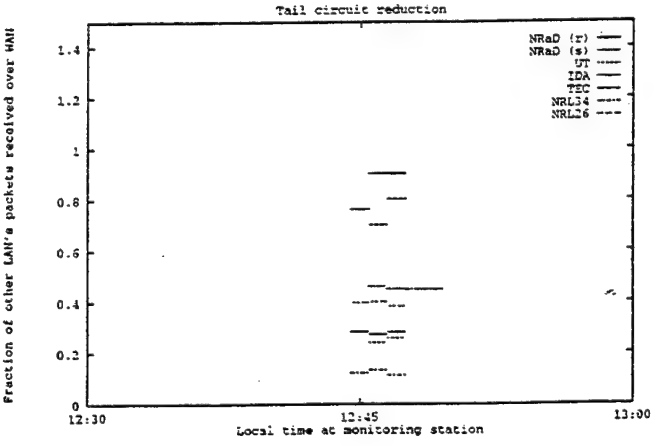






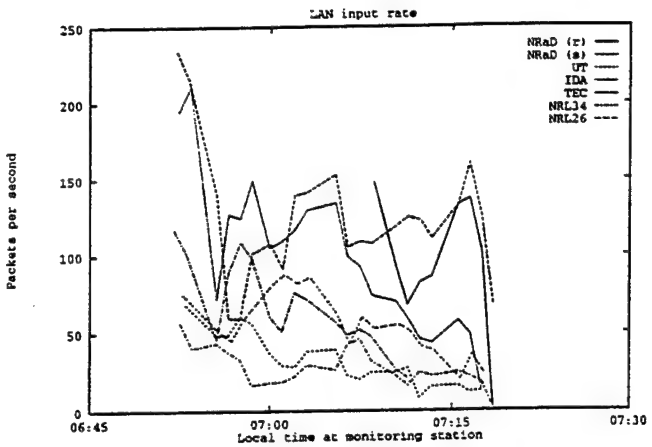


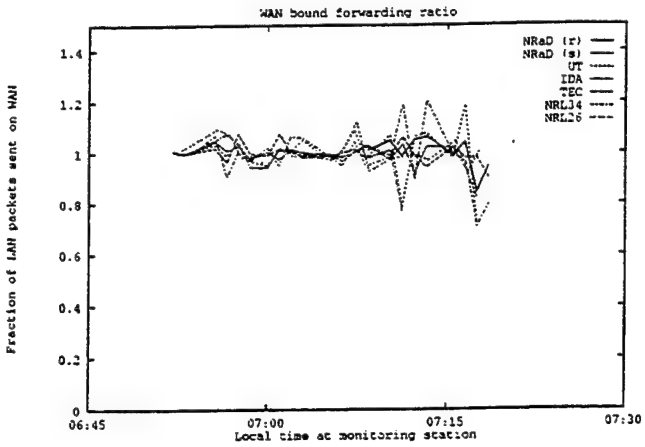


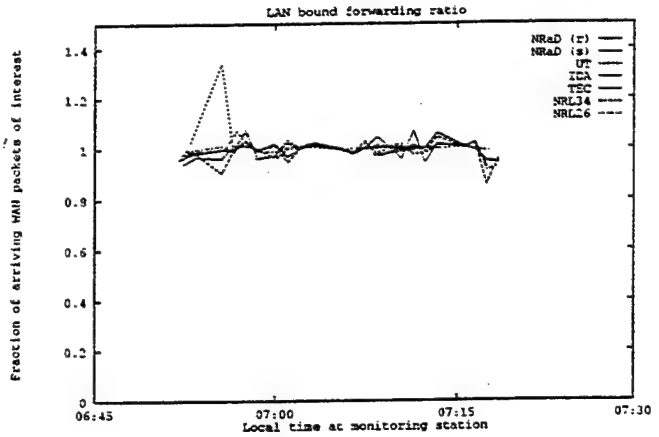


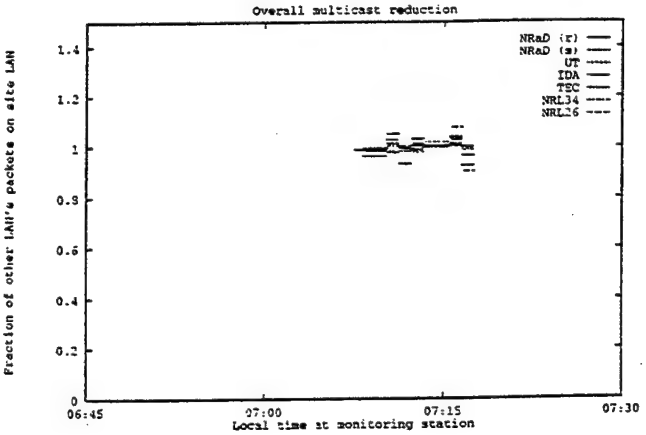
Run on 14-Nov-95 data from 12:30 to 13:00
Large Scenario
no QES
WAN multicast, LAN multicast

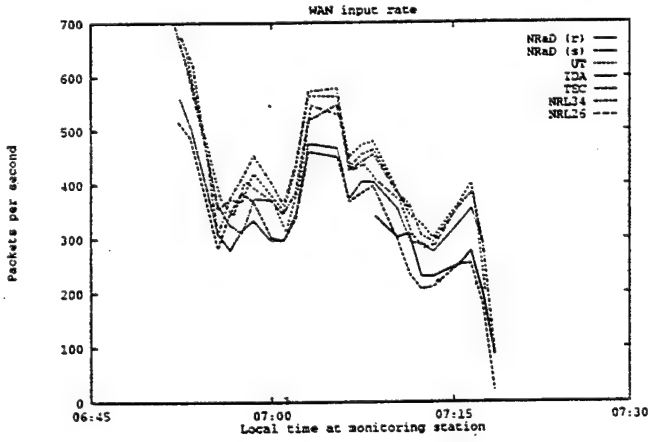
Event 3

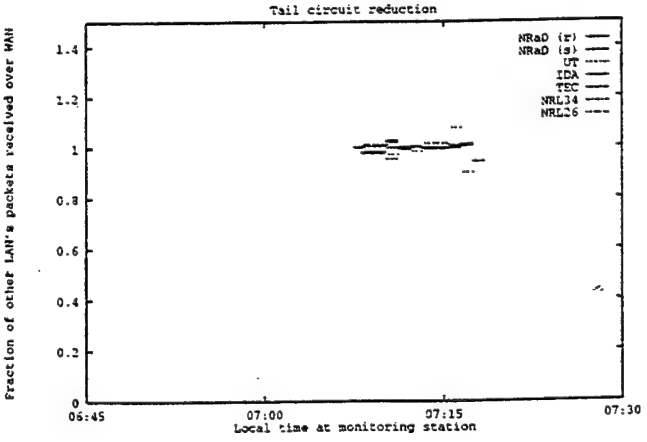






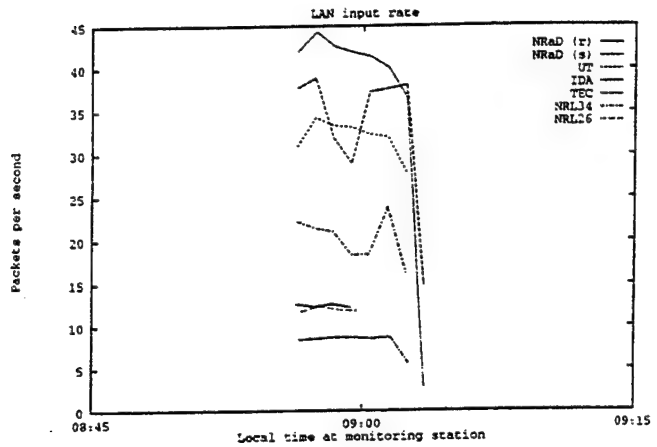


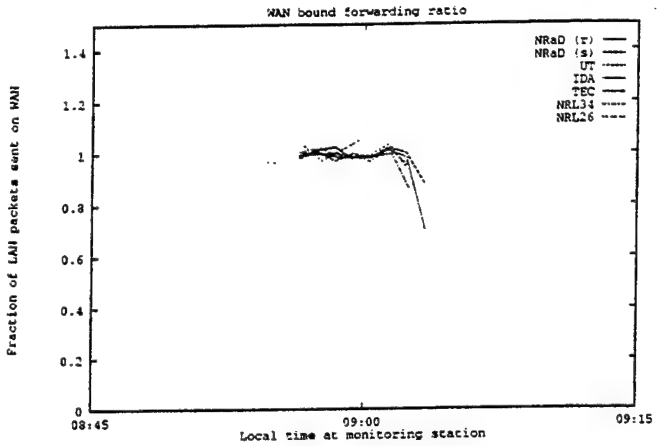


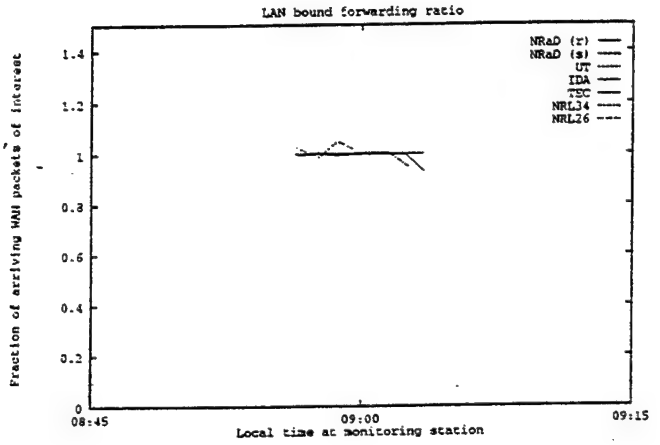


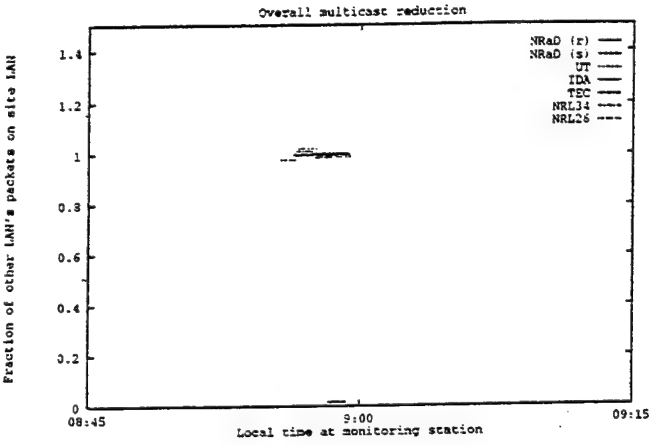
Run on 15-Nov-95 data from 6:45 to 7:30
Large Scenario
QES only
no WAN multicast, no LAN multicast

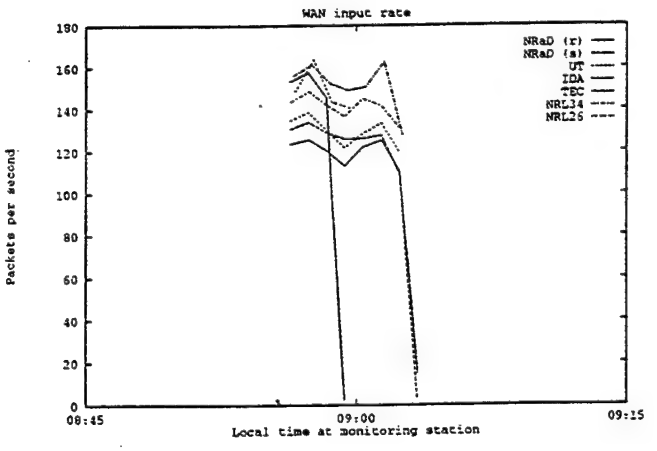
Event 4

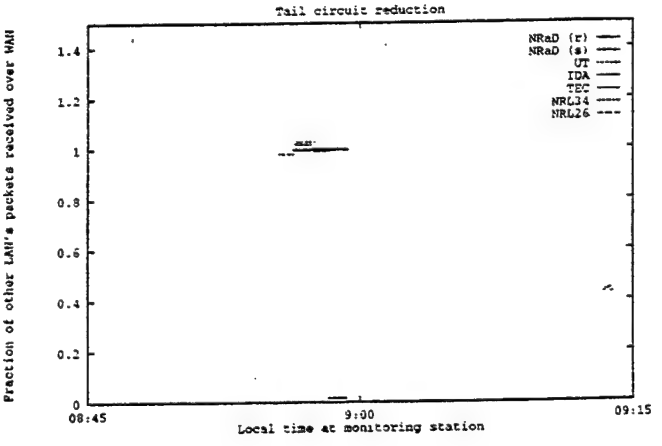






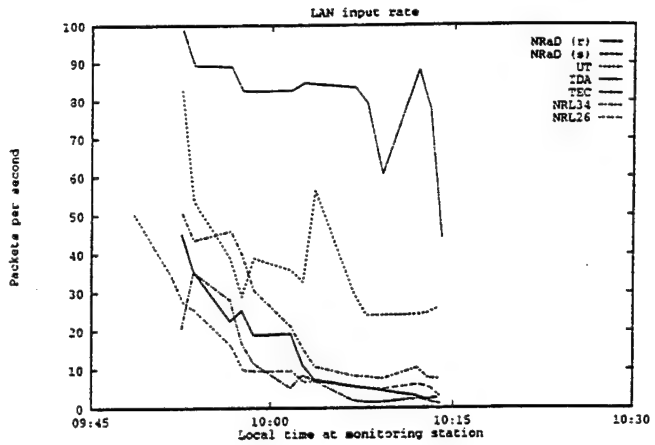


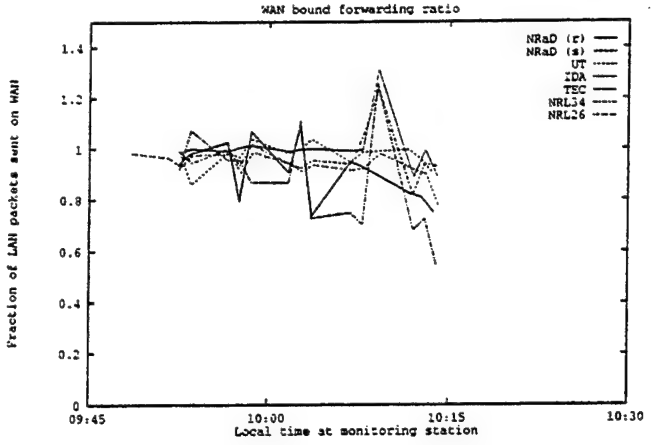


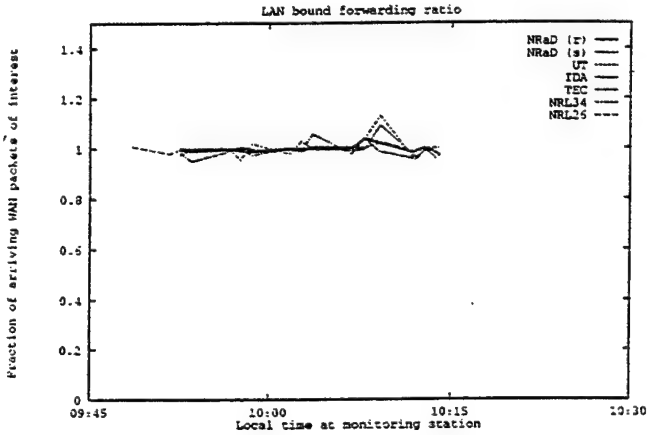


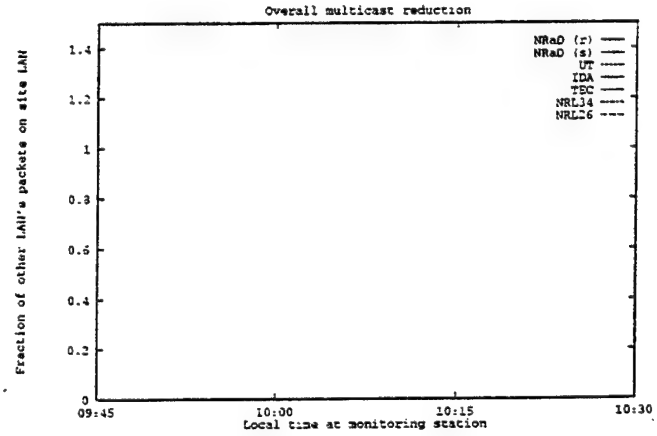
Run on 16-Nov-95 data from 8:45 to 9:15
Small Scenario
no ACT algorithms
no WAN multicast, no LAN multicast

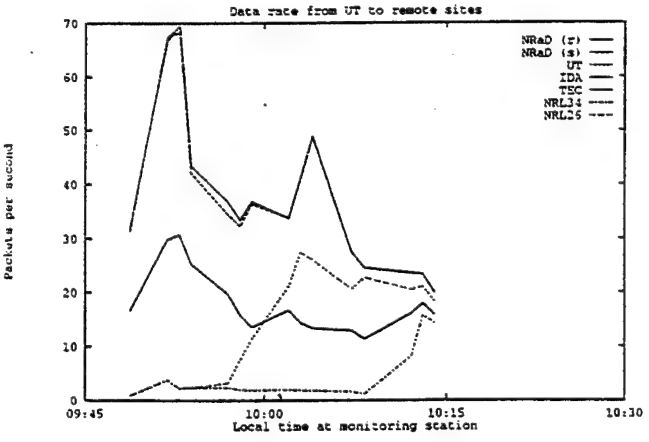
Event 6

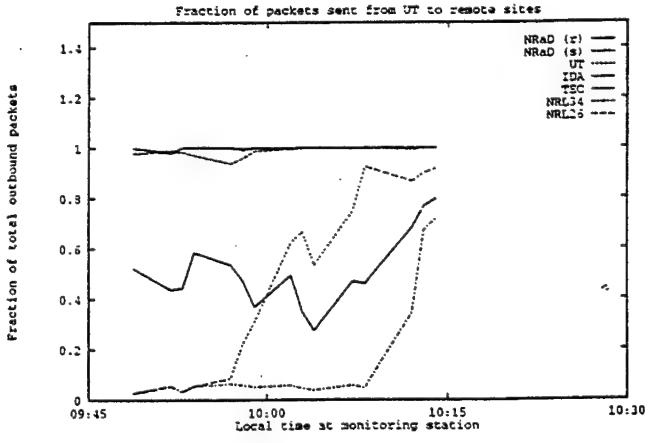






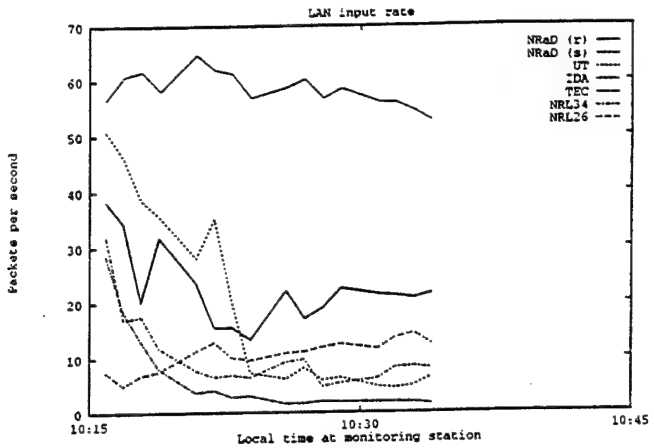


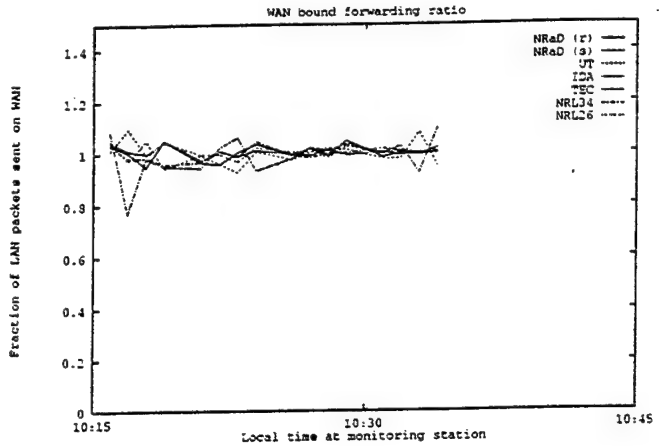


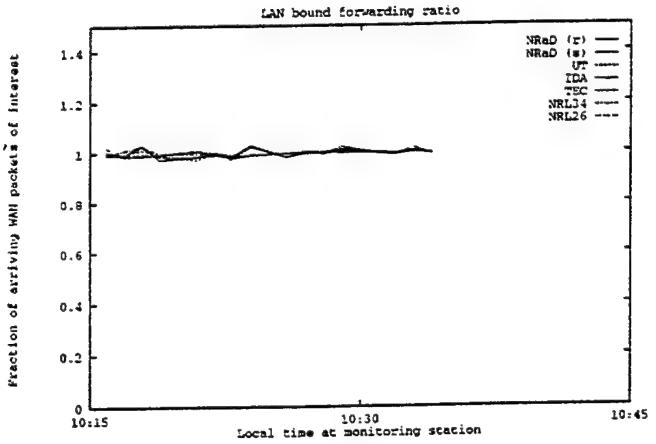


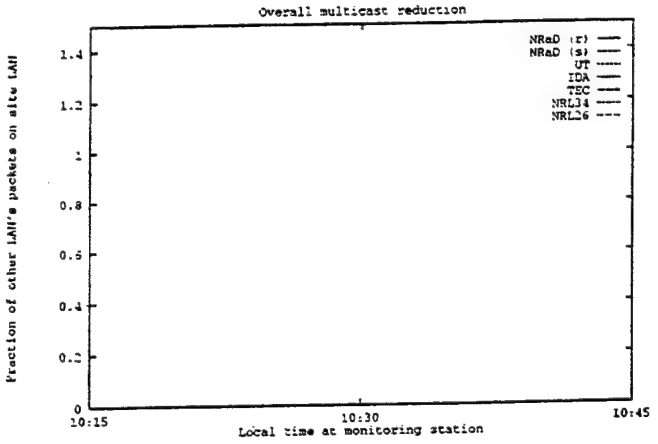
Run on 16-Nov-95 data from 9:45 to 10:30
Small Scenario
All Algorithms
WAN multicast, LAN multicast

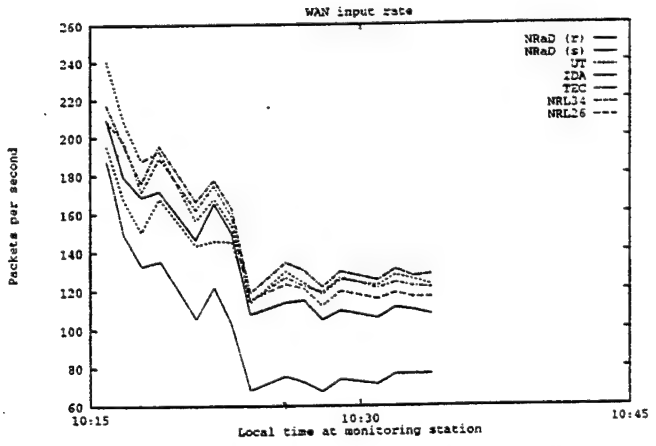
Event 7

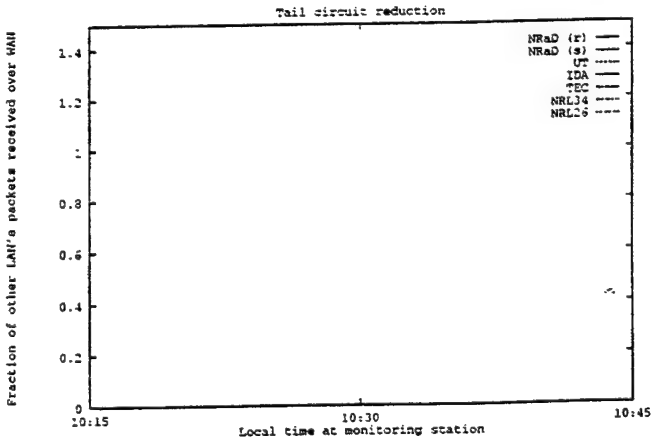






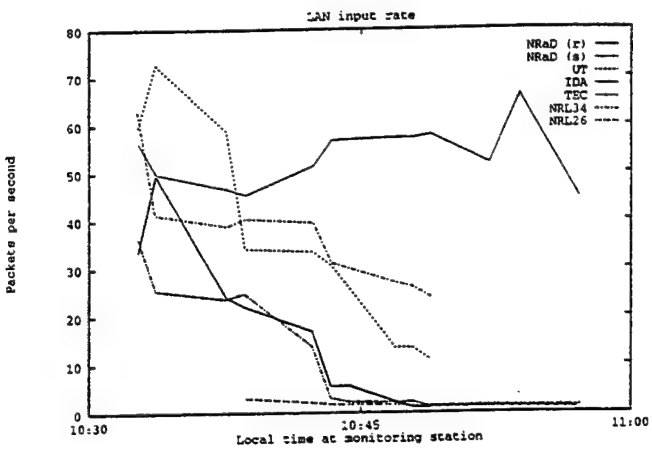


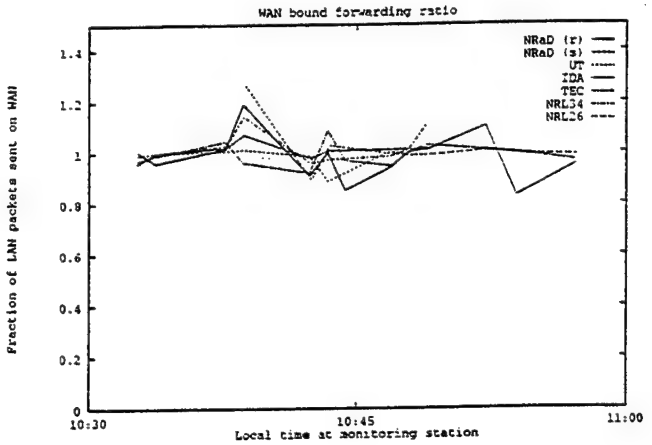


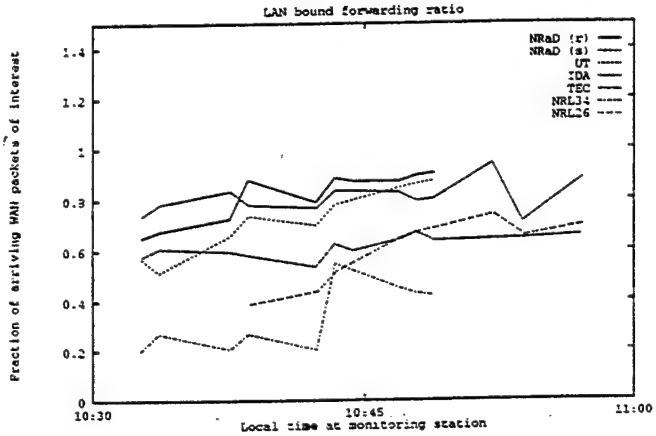


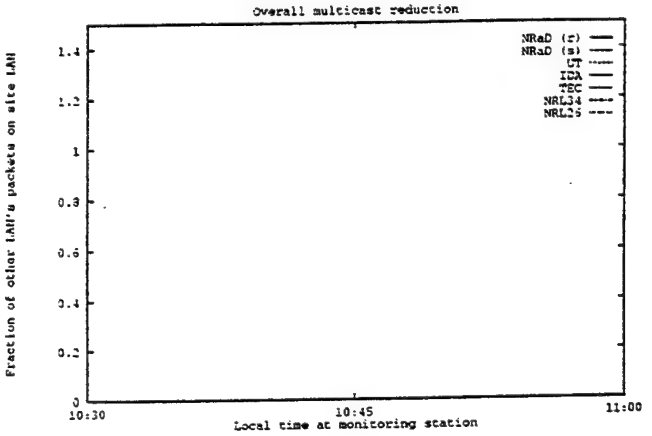
Run on 15-Nov-95 data from 10:15 to 10:45
Small Scenario
QES only
no WAN multicast, no LAN multicast

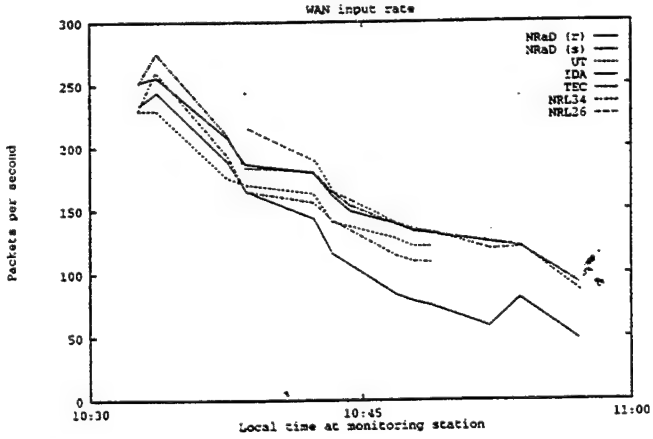
Event 8

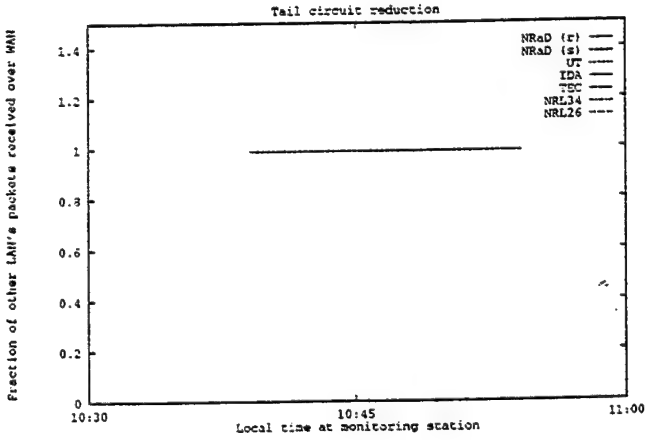






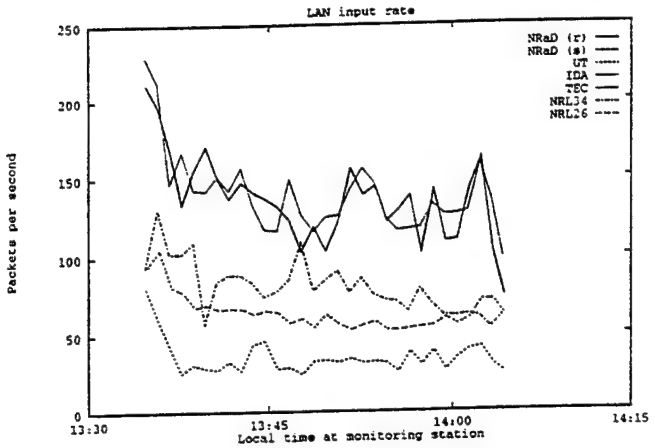


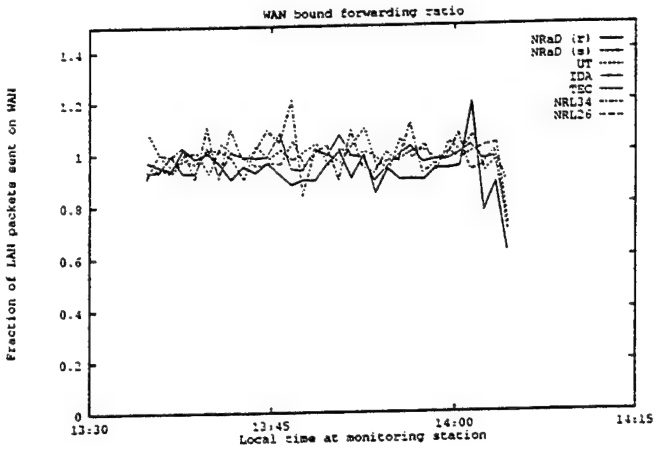


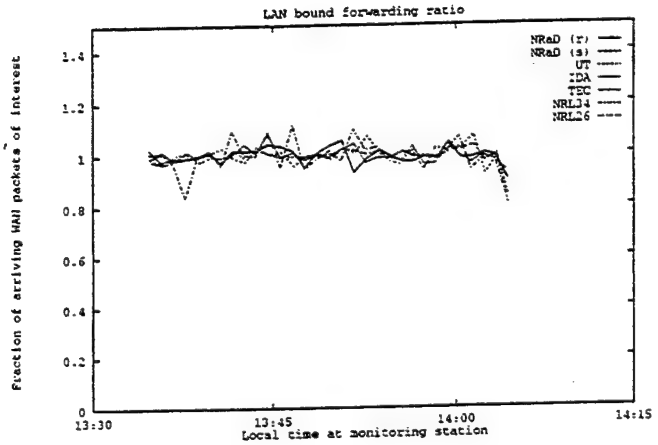


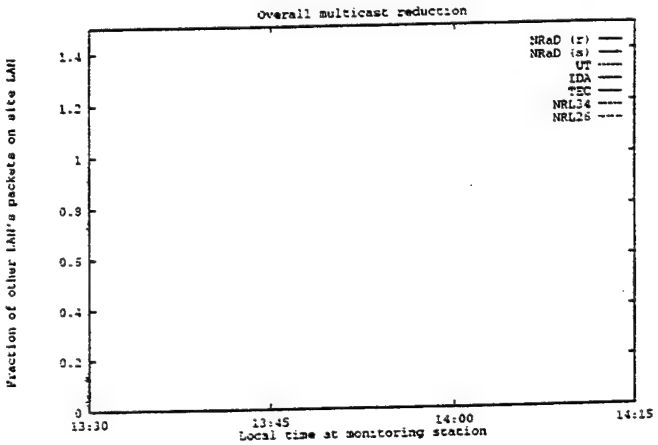
Run on 16-Nov-95 data from 10:30 to 11:00
Small Scenario
QES, LM
no WAN multicast, LAN multicast

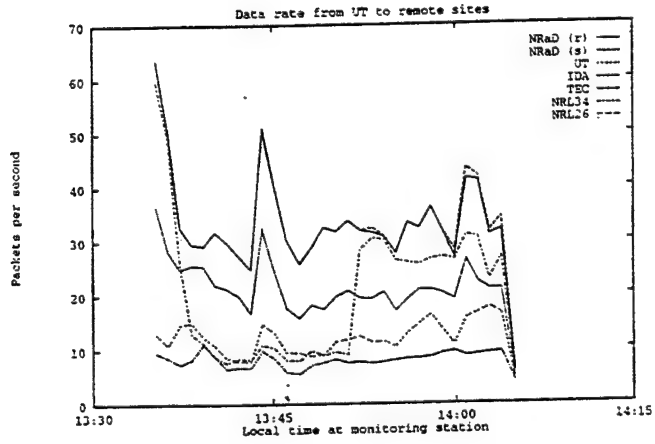
Event 9

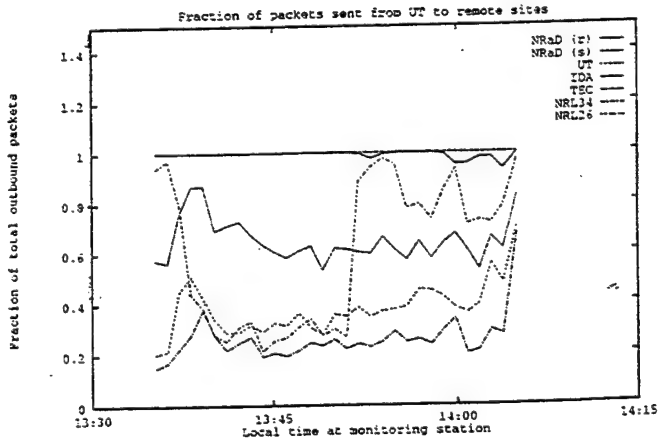






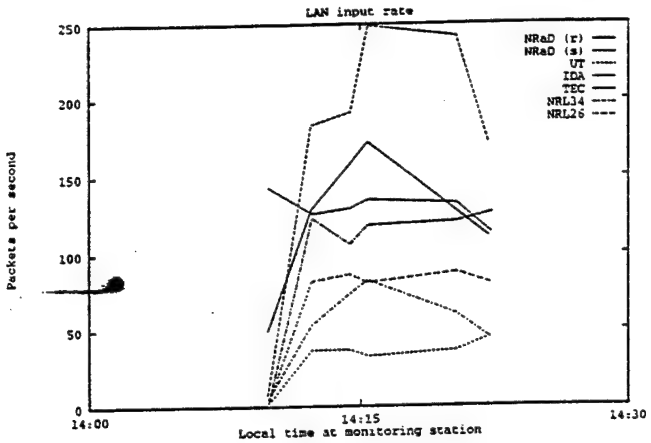


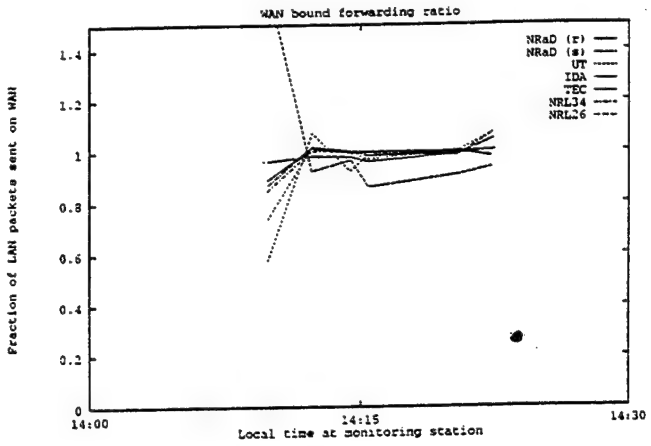


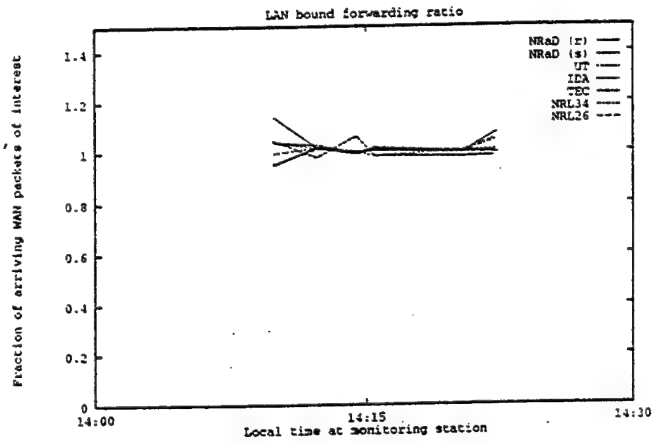


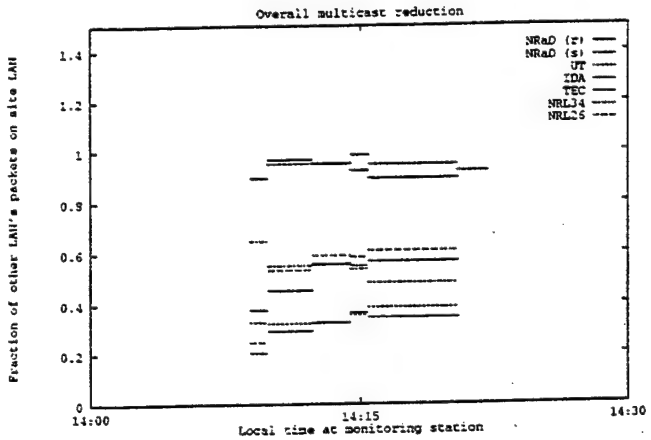
Run on 16-Nov-95 data from 13:30 to 14:15
Large Scenario
All Algorithms
WAN multicast, Alt3 LAN multicast

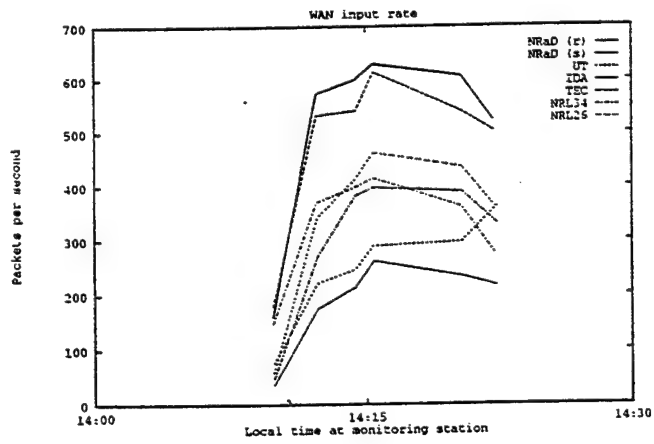
Event 10

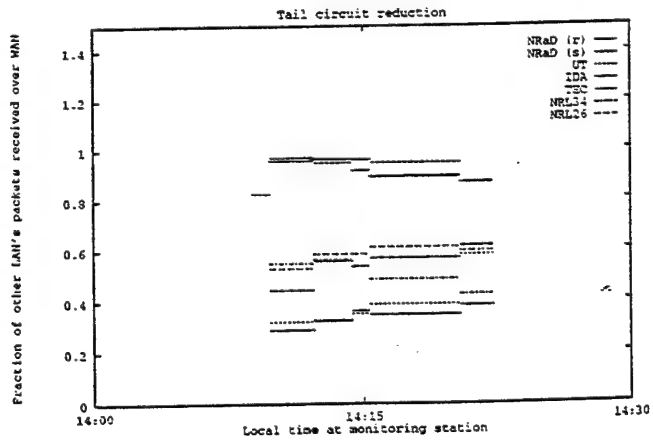






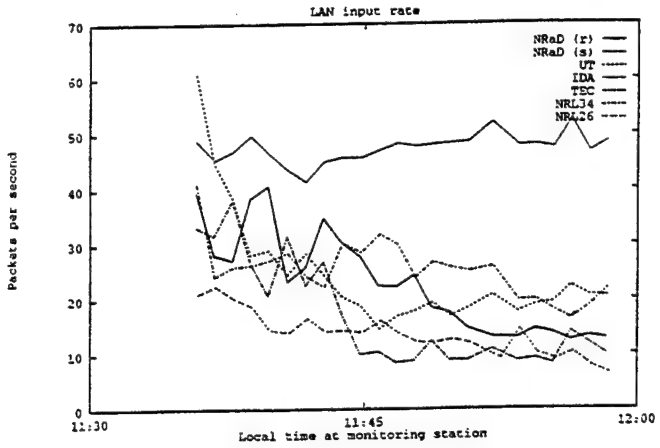


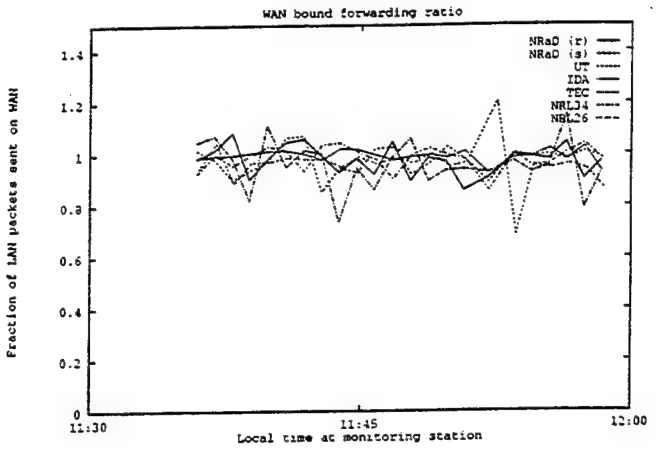


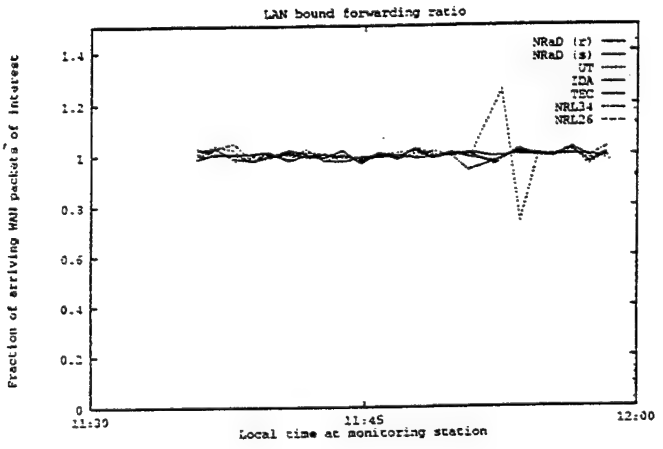


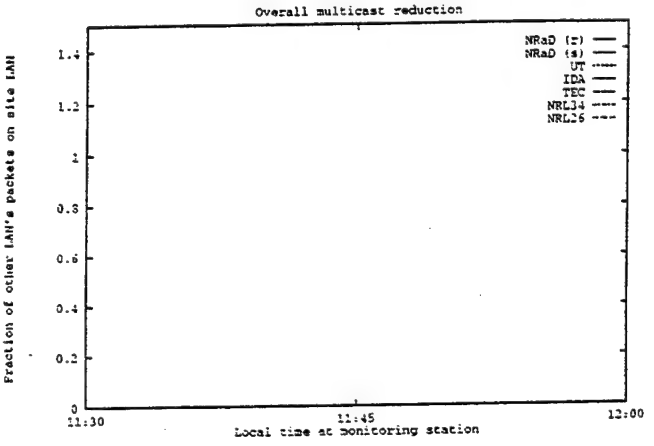
Run on 15-Nov-95 data from 14:00 to 14:30
Large Scenario
All Algorithms
WAN multicast, Alt2 LAN multicast

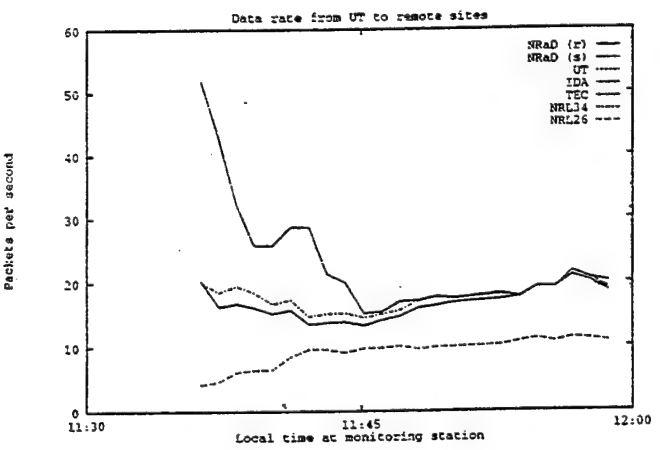
Event 11

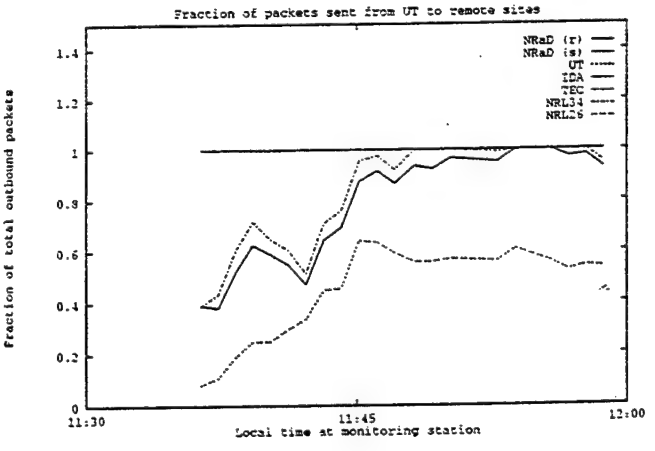






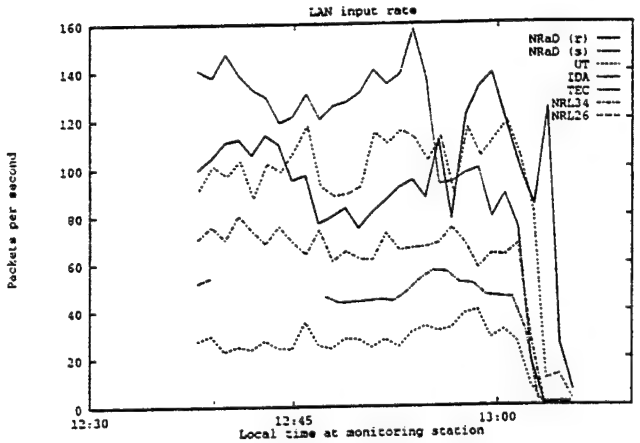


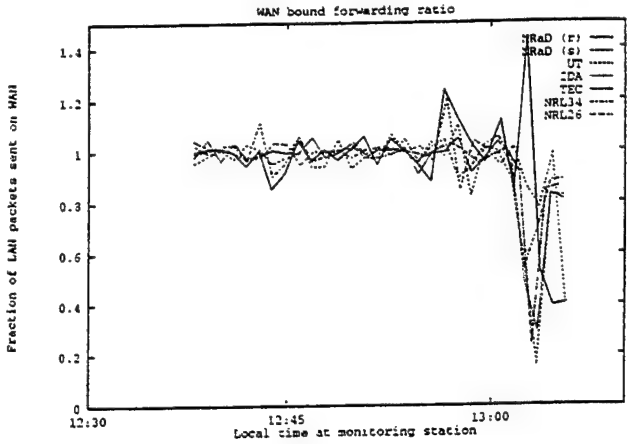


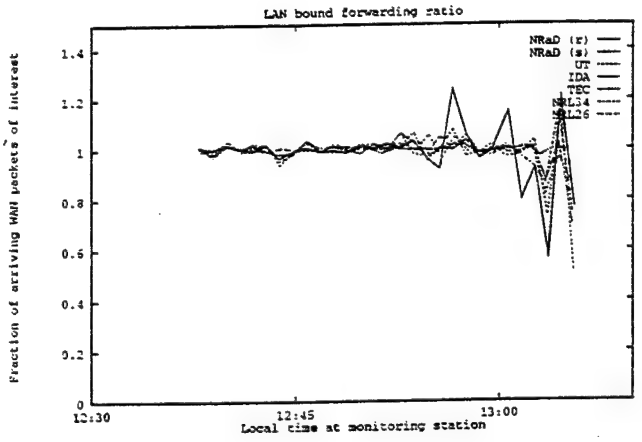


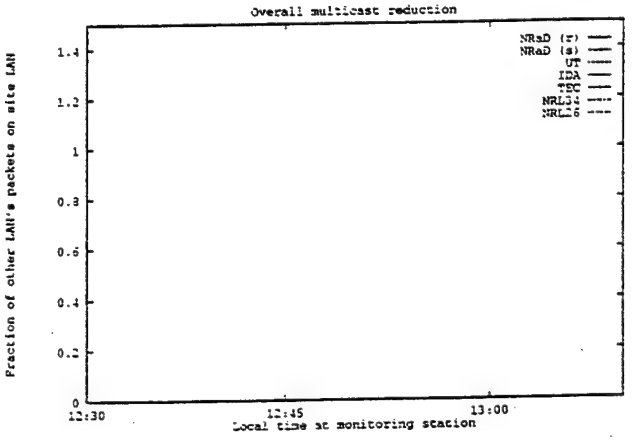
Run on 16-Nov-95 data from 11:30 to 12:00
Synthetic Environment in Small Scenario
All Algorithms
WAN multicast, LAN multicast

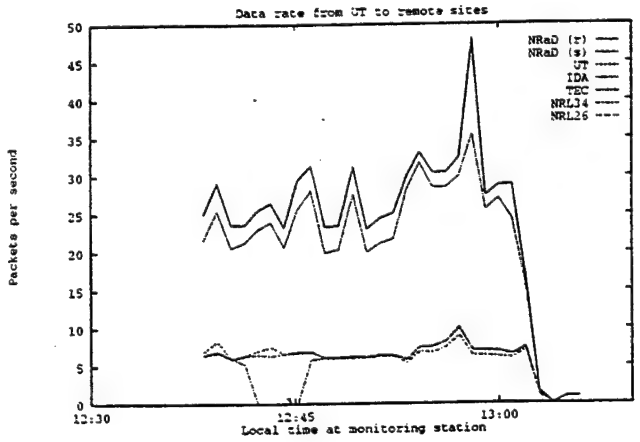
Event 13

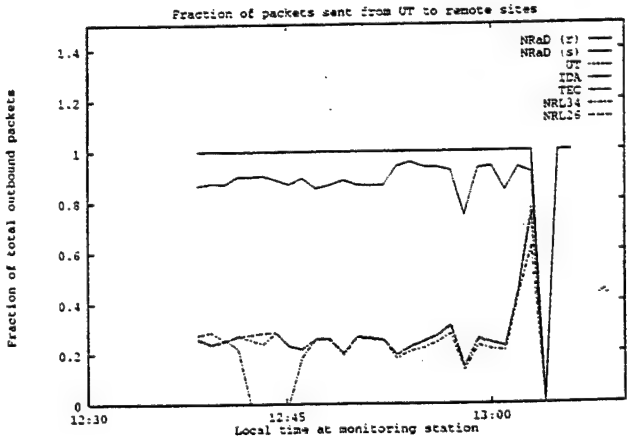












Run on 16-Nov-95 data from 12:30 to 13:10
Large Scenario
All Algorithms
WAN multicast, LAN multicast

Event max

APPENDIX C -

HIGH PERFORMANCE APPLICATION GATEWAY (HPAG) MIB

```
--
**************************************************************************
--
--   1.0    95/04/14 R. K. Nair
--          High Performance Application Gateway(HPAG) MIB
--
--   1.2    95/10/27 R. K. Nair
--      New variables added to the MIB for ED1 A
--
**************************************************************************
/


HPAG-MIB DEFINITIONS ::= BEGIN

    IMPORTS      Counter
                    FROM RFC1155-SMI
                 OBJECT-TYPE
                    FROM RFC-1212
                 DisplayString
                    FROM RFC1213-MIB
                 TRAP-TYPE
                    FROM RFC1215;

    nrl  OBJECT IDENTIFIER ::= { enterprises 394 }
    ritn OBJECT IDENTIFIER ::= { nrl 2 }
    hpag   OBJECT IDENTIFIER ::= { ritn 1 }
    ati  OBJECT IDENTIFIER ::= { hpag 2 }

    -- The system group contains general information about the
    -- application software.

    sysDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-only
        STATUS  mandatory
            DESCRIPTION
                "A textual description of the entity.  This value
             should include the full name and version
                identification of the application software,
             software operating-system, and other
                descriptive text. It is mandatory that this only contain
                printable ASCII characters."
            ::= { ati 1 }

    sysContact OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-write
            STATUS  mandatory
```

```
        DESCRIPTION
            "The textual identification of the contact person
            for this application software, together with information
            on how to contact this person."
        ::= { ati 2 }

laninpktssec    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "LAN input packets per second"
    ::= { ati 3 }

lanoutpktssec  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "LAN output packets per second"
    ::= { ati 4 }

laninkbitssec   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "LAN # kbits input/second"
    ::= { ati 5 }

lanoutkbitssec  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "LAN # kibits output/second"
    ::= { ati 6 }

waninpktssec  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "WAN input packets per second"
    ::= { ati 7 }

wanoutpktssec OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "WAN output packets per second"
    ::= { ati 8 }
```

```
waninkbitssec OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "WAN # kbits input/second"
    ::= { ati 9 }

wanoutkbitssec    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "WAN # kibits output/second"
    ::= { ati 10 }

pducount  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of PDUs by kind"
    ::= { ati 11 }

llstatus OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "load leveling status"
    ::= { ati 12 }

kbsec      OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "ll kibs per second output"
    ::= { ati 13 }

pktsdropped   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "number of dropped packets"
    ::= { ati 14 }

bytesdropped  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "number of dropped bytes"
```

```
    ::= { ati 15 }

pktsforwarded OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "good packets sent on"
    ::= { ati 16 }

bundlingstatus OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "bundling ON/OFF status"
    ::= { ati 17 }

bundledelay   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "ms delay for bundling"
    ::= { ati 18 }

avepktsbundle OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "average packets per bundle"
    ::= { ati 19 }

tempint1   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 20 }

tempint2   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 21 }
```

```
tempint3    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 22 }

tempint4    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 23 }

tempint5    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 24 }

tempint6    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 25 }

tempint7    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 26 }

tempint8    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
```

```
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 27 }

tempint9    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 28 }

tempint10  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 29 }

tempint11  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 30 }

tempint12  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 31 }

tempint13  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
```

```
    ::= { ati 32 }

tempint14  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 33 }

tempint15  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 34 }

tempint16  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 35 }

tempint17  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 36 }

tempint18  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 37 }

tempint19  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
```

```
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ati 38 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ati 39 }

    overloadtrap   OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "ll overload condition"
        ::= { ati 40 }

    overloadcondition  TRAP-TYPE
        ENTERPRISE ati
        VARIABLES  { overloadtrap }
        DESCRIPTION
            "ll overload condition"
        ::= 0

END
--
*************************************************************************
--
--   1.0     95/04/26 R. K. Nair
--           High Performance Application Gateway(HPAG) MIB - Multicast
--      specific variables.
--
--   1.1     95/10/06 R. K. Nair
--      HPAG MIB - MC Section Update for the ED1 Demo
--
--   1.2     95/10/27 R. K. Nair
--      HPAG MIB - MC Section Update for the ED1 A Demo
--
--
*************************************************************************
/


HPAG-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS      Counter
                FROM RFC1155-SMI
             OBJECT-TYPE
                FROM RFC-1212
             DisplayString
                FROM RFC1213-MIB
             TRAP-TYPE
                FROM RFC1215;

nrl  OBJECT IDENTIFIER ::= { enterprises 394 }
ritn OBJECT IDENTIFIER ::= { nrl 2 }
hpag   OBJECT IDENTIFIER ::= { ritn 1 }
mc OBJECT IDENTIFIER ::= { hpag 3 }

-- The system group contains general information about the
-- application software.

sysDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
        DESCRIPTION
            "A textual description of the entity.  This value
         should include the full name and version
            identification of the application software,
         software operating-system, and other
            descriptive text. It is mandatory that this only contain
            printable ASCII characters."
        ::= { mc 1 }

sysContact OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The textual identification of the contact person
            for this application software, together with information
            on how to contact this person."
        ::= { mc 2 }

joinLeaveHoldoff  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS   read-write
    STATUS   mandatory
    DESCRIPTION
        "control var - JOIN/LEAVE hold-off time.
        time HPAG will wait to send next join/leave
        Minimum time between join msgs."
    ::= { mc 3 }

ackInterval OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS   read-write
```

```
    STATUS  mandatory
    DESCRIPTION
        "control for bi-level mc protocol"
    ::= { mc 4 }

stateIntervalTransfer  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "control variable - time interval between all of
        the packets containing state information for a
        new HPAG.  If a machine reboots, and needs all
        state information"
    ::= { mc 5 }

stateIntervalHash  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "control variable - longer time interval than above
        time between pkts which are sent in response to
        hash table error"
    ::= { mc 6 }

msgsWithoutAck  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Number of messages without ack"
    ::= { mc 7 }

timeWithoutAck  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Elapsed Time Without ack"
    ::= { mc 8 }


mcGroupRangeSize  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 9 }

mcGroupRangeTable  OBJECT-TYPE
    SYNTAX  SEQUENCE OF MCGroupRangeTableEntry
    ACCESS  not-accessible
```

```
    STATUS  mandatory
    ::= { mc 10 }


mcGroupRangeTableEntry    OBJECT-TYPE
    SYNTAX  MCGroupRangeTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { mcGroupRangeIndex }
    ::= { mcGroupRangeTable 1 }

MCGroupRangeTableEntry ::= SEQUENCE { mcGroupRangeIndex INTEGER,
                lowAddress INTEGER,
                highAddress INTEGER }

lowAddress    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mcGroupRangeTableEntry 1 }

highAddress   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mcGroupRangeTableEntry 2 }

mcGroupRangeIndex  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mcGroupRangeTableEntry 3 }

numberJoins   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Total number of LAN groups joined"
    ::= { mc 11 }

numberLeaves OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Total number of LAN groups left"
    ::= { mc 12 }
```

```
peerStatusSize OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 13 }

peerStatusTable    OBJECT-TYPE
    SYNTAX  SEQUENCE OF PeerStatusTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { mc 14 }

peerStatusTableEntry  OBJECT-TYPE
    SYNTAX  PeerStatusTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX     { peerStatusTableIndex }
    ::= { peerStatusTable 1 }

PeerStatusTableEntry ::= SEQUENCE { peerStatusTableIndex INTEGER,
                peerAddress1 INTEGER,
                ackLocalSequence INTEGER,
                ackRemoteSequence INTEGER,
                bootControlPackets INTEGER,
                ackControlPackets INTEGER,
                jlControlPackets INTEGER,
                ddControlPackets INTEGER,
                hashControlPackets INTEGER,
                rsControlPackets INTEGER,
                stateControlPackets INTEGER,
                misorderedPackets INTEGER,
                retransmitCaused INTEGER,
                stateErrors INTEGER,
                ottToTypeTime INTEGER,
                ottFromTypeTime INTEGER }

peerAddress1  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 1 }

ackLocalSequence OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 2 }
```

```
ackRemoteSequence   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 3 }

bootControlPackets    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 4 }

ackControlPackets OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 5 }

jlControlPackets    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 6 }

ddControlPackets  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 7 }

hashControlPackets    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "",
    ::= { peerStatusTableEntry 8 }

rsControlPackets  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
```

```
        ""
    ::= { peerStatusTableEntry 9 }

stateControlPackets    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 10 }

misorderedPackets OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 11 }

retransmitCaused  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 12 }

stateErrors OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 13 }

ottToTypeTime    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 14 }

ottFromTypeTime OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 15 }

peerStatusTableIndex  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
```

```
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { peerStatusTableEntry 16 }

mcPacketsRcvdOnLAN   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of MC Packets Received on LAN
        by the HPAG"
    ::= { mc 15 }

mcBytesRcvdOnLAN OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of MC Bytes Received on LAN
        by the HPAG"
    ::= { mc 16 }

routingStatusSize  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 17 }

routingStatusTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF RoutingStatusTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { mc 18 }

routingStatusTableEntry   OBJECT-TYPE
    SYNTAX  RoutingStatusTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { routingStatusTableIndex }
    ::= { routingStatusTable 1 }

RoutingStatusTableEntry ::= SEQUENCE { routingStatusTableIndex INTEGER,
                ipAddress INTEGER,
                peerAddress2 INTEGER }

ipAddress  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
```

```
    ::= { routingStatusTableEntry 1 }

peerAddress2  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { routingStatusTableEntry 2 }

routingStatusTableIndex   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { routingStatusTableEntry 3 }

wanMCPacketsRcvd  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of WAN MC Packets Received"
    ::= { mc 19 }

wanMCBytesRcvd    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of WAN MC Bytes Received"
    ::= { mc 20 }

wanMCPacketsDelivered  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of WAN MC Packets Delivered"
    ::= { mc 21 }

wanMCBytesDelivered    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Number of WAN MC Bytes Delivered"
    ::= { mc 22 }

igmpTableSize OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
```

```
    DESCRIPTION
        ""
    ::= { mc 23 }

igmpTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IGMPTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { mc 24 }

igmpTableEntry    OBJECT-TYPE
    SYNTAX  IGMPTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX     { igmpTableIndex }
    ::= { igmpTable 1 }

IGMPTableEntry ::= SEQUENCE { igmpTableIndex INTEGER,
                group INTEGER,
                timeToLive INTEGER }

group  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { igmpTableEntry 1 }

timeToLive     OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { igmpTableEntry 2 }

igmpTableIndex    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { igmpTableEntry 3 }

bmppReconfig OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if set to 1, check rest of snmp state, clear"
    ::= { mc 25 }

bmppForward OBJECT-TYPE
```

```
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if set to 1, forward packets"
    ::= { mc 26 }

bmppDeliverAlways   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if set to 1, deliver even if not joined"
    ::= { mc 27 }

bmppSendAllSites OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if set to 1, always forward on allsites group"
    ::= { mc 28 }

bmppNoProtocol  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if set to 1, don't do protocol exchange"
    ::= { mc 29 }

bmppProtoTTL    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "ttl to use for protocol packets"
    ::= { mc 30 }

bmppDataTTL OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "ttl to use for data packets"
    ::= { mc 31 }

bmppRandomDropOutput OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "To Satisfy, Dan Van Hook's Testing Needs,
        drop (n out of 65535) output data packets,
```

```
            on average, randomly"
    ::= { mc 32 }

bmppRandomDropInput   OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "To Satisfy, Dan Van Hook's Testing Needs,
         drop (n out of 65535) input data packets,
            on average, randomly"
    ::= { mc 33 }

bmppOutputError  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 34 }

bmppParseError   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "input parse failed on packet"
    ::= { mc 35 }

bmppEmptyPacketOBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "no messages in packet"
    ::= { mc 36 }

bmppWrongVersion   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "invalid version"
    ::= { mc 37 }

bmppUnknownType  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "invalid type"
    ::= { mc 38 }

bmppHashMatch  OBJECT-TYPE
```

```
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "crypto checksum matches ok"
    ::= { mc 39 }

bmppHashMismatch  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "crypto checksum does not match ok"
    ::= { mc 40 }

bmppOttNegative  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "negative one-way trip time"
    ::= { mc 41 }

bmppOttHuge  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "huge negative one-way trip time"
    ::= { mc 42 }

bmppMsgBootReceived  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "boot messages received"
    ::= { mc 43 }

bmppMsgAckReceived  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "ack messages received"
    ::= { mc 44 }

bmppMsgQueueReceived  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "messages received that get put in the
        queue (not boot, ack)"
```

```
    ::= { mc 45 }

bmppMsgQueueDups OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "of messages fro queue, duplicate
         messages"
    ::= { mc 46 }

bmppSeqOutOfBounds    OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "received sequence number unreasonable"
    ::= { mc 47 }

bmppHeardOutOfBounds OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "heard sequence number unreasonable"
    ::= { mc 48 }

stateStartRequests OBJECT-TYPE
    SYNTAX INTEGER (0..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "count of times requested to send state"
    ::= { mc 49 }

bmppStateSend    OBJECT-TYPE
    SYNTAX INTEGER (0..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "count of starting to send state sequence"
    ::= { mc 50 }

bmppStateResend OBJECT-TYPE
    SYNTAX INTEGER (0..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "count of restarting"
    ::= { mc 51 }

bmppStateIdle OBJECT-TYPE
    SYNTAX INTEGER (0..2147483647)
    ACCESS read-only
```

```
        STATUS  mandatory
        DESCRIPTION
            "count of finishing"
        ::= { mc 52 }

    bmppStateShort    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "incoming state messages that are too short"
        ::= { mc 53 }

    bmppStateGroupUnknown    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "incoming state messages with unknown groups"
        ::= { mc 54 }

    bmppStateEarlyZero   OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "incoming state messages with syntax errors"
        ::= { mc 55 }

    bmppStateOK  OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "count of ok state checks"
        ::= { mc 56 }

    bmppStateError    OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "count of state messages with errors"
        ::= { mc 57 }

    bmppStateErrorCount OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "number of mismatched groups"
        ::= { mc 58 }

    bmppStateTransfer     OBJECT-TYPE
```

```
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "count of state messages that don't
             match when it is ok"
        ::= { mc 59 }

    bmppStateTransferCount  OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "number of mismatched groups"
        ::= { mc 60 }

    igmpQueryInterval OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { mc 61 }

    igmpImpliedLeave OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { mc 62 }

    igmpOutputError  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { mc 63 }

    igmpIPParseError OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { mc 64 }

    igmpNotIgmp  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            ""
```

```
    ::= { mc 65 }

igmpIgmpParseError  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 66 }

igmpQueriesSent  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "IGMP queries transmitted to the
        application"
    ::= { mc 67 }

igmpQueriesReceived  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 68 }

igmpWrongAddr  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 69 }

igmpReportNotHandled  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 70 }

igmpUnknownType  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 71 }

igmpCallback  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
```

```
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 72 }

bmppInsufficientLanTTL  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Packets which are dropped because their
        time to live has been exceeded (LAN side).
        This is an Error Condition"
    ::= { mc 73 }

bmppInsufficientWanTTL OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Packets which are dropped because their
        time to live has been exceeded (WAN side).
        This is an Error Condition"
    ::= { mc 74 }

bmppLANIntentionalDrop OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Packets which are dropped because no other
        site is requesting that mc group"
    ::= { mc 75 }

bmppWANIntentionalDrop    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Packets which are dropped because local
        simulations are not interested.  This is
        LAN savings due to multicast"
    ::= { mc 76 }

bmppRandomDropControlInput  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Debug Option to control Random Drop"
    ::= { mc 77 }

bmppControlInputIntentionalDrop    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
```

```
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Control Intentional Drop"
    ::= { mc 78 }

bmppRexmitMsg  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 79 }

bmppRexmitPacket    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { mc 80 }

bmeOLMXmitPacket  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Good MC packets, which are forwarded
        to the overload management module"
    ::= { mc 81 }

bmppOtherDrop    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Packets dropped due to other Errors"
    ::= { mc 82 }

igmpOKReportsRcvd  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Error free IGMP reports received"
    ::= { mc 83 }

bmppInCntlPacket OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Control Packets Received from
        other HPAGs"
```

```
    ::= { mc 84 }

bmppInCntlBytes  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Total number of bytes in the Control
        Packets Received from other HPAGs.
        This includes UDP/IP headers, not link
        layer"
    ::= { mc 85 }

igmpInPackets OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "IGMP packets Rcvd"
    ::= { mc 86 }

tempint11  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 87 }

tempint12  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 88 }

tempint13  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 89 }

tempint14  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
```

```
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 90 }

tempint15  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 91 }

tempint16  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 92 }

tempint17  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 93 }

tempint18  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 94 }

tempint19  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
```

```
        list on the fly"
    ::= { mc 95 }

tempint20  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 96 }

tempint21  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 97 }

tempint22  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 98 }

tempint23  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 99 }

tempint24  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 100 }

tempint25  OBJECT-TYPE
```

```
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 101 }

tempint26  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 102 }

tempint27  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 103 }

tempint28  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 104 }

tempint29  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { mc 105 }

tempint30  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
```

```
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { mc 106 }

    rk1 OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { mc 107 }

    bob1   OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { mc 108 }
END
--
****************************************************************************
--
--   1.0    95/05/01 R. K. Nair
--          The WAN Interface MIB - A part of AH Software
--
--   1.1    95/10/09 R. K. Nair
--      ED1 modifications to the WAN Interface MIB
--
--   1.2    95/10/27 R. K. Nair
--      ED1 A modifications to the WAN Interface MIB
--
--
****************************************************************************
/


HPAG-WANINT-MIB DEFINITIONS ::= BEGIN

    IMPORTS      Counter
                    FROM RFC1155-SMI
                OBJECT-TYPE
                    FROM RFC-1212
                DisplayString
                    FROM RFC1213-MIB
                TRAP-TYPE
                    FROM RFC1215;
```

```
nrl OBJECT IDENTIFIER ::= { enterprises 394 }
ritn OBJECT IDENTIFIER ::= { nrl 2 }
hpag    OBJECT IDENTIFIER ::= { ritn 1 }
wanint OBJECT IDENTIFIER ::= { hpag 4 }

-- The system group contains general information about the
-- application software.

sysDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
        DESCRIPTION
            "A textual description of the entity.  This value
         should include the full name and version
            identification of the application software,
         software operating-system, and other
            descriptive text. It is mandatory that this only contain
            printable ASCII characters."
        ::= { wanint 1 }

sysContact OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The textual identification of the contact person
            for this application software, together with information
            on how to contact this person."
        ::= { wanint 2 }

configState OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Used to initiate WAN interface.
         0 - not initiated
         1 - go initiate
         2 - complete"
    ::= { wanint 3 }

wanType         OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "currently is IPv4"
    ::= { wanint 4 }

wanSetupAddrSize    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
```

```
    DESCRIPTION
        "number of bytes in an address to init
        connection - currently is 4"
    ::= { wanint 5 }

wanDataAddrSize OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "number of bytes in an address to init
        connection - currently is 4"
    ::= { wanint 6 }

numberWANGroups  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Default is 1024 - a filled in control
        parameter"
    ::= { wanint 7 }

numberSites   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "number of HPAGS = number of sites"
    ::= { wanint 8 }

siteListTable   OBJECT-TYPE
    SYNTAX  SEQUENCE OF SiteListTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { wanint 9 }

siteListTableEntry OBJECT-TYPE
    SYNTAX  SiteListTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { siteIndex }
    ::= { siteListTable 1 }

SiteListTableEntry ::= SEQUENCE { siteIndex INTEGER,
                siteList INTEGER }

siteList OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "IP Address of HPAGs"
    ::= { siteListTableEntry 1 }
```

```
siteIndex   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "site id"
    ::= { siteListTableEntry 2 }

wanGroupBaseAddrTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF WANGroupBaseAddrTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { wanint 10 }

wanGroupBaseAddrTableEntry   OBJECT-TYPE
    SYNTAX  WANGroupBaseAddrTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { wanGroupIndex }
    ::= { wanGroupBaseAddrTable 1 }

WANGroupBaseAddrTableEntry ::= SEQUENCE { wanGroupIndex INTEGER,
            wanGroupBaseAddr INTEGER }

wanGroupBaseAddr   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "first MC address to use on WAN"
    ::= { wanGroupBaseAddrTableEntry 1 }

wanGroupIndex   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "WAN group id"
    ::= { wanGroupBaseAddrTableEntry 2 }

bwMultiplierTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF BWMultiplierTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { wanint 11 }

bwMultiplierTableEntry   OBJECT-TYPE
    SYNTAX  BWMultiplierTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { bwMultiplierTableIndex }
    ::= { bwMultiplierTable 1 }
```

```
BWMultiplierTableEntry ::= SEQUENCE { bwMultiplierTableIndex INTEGER,
            bwMultiplier INTEGER }

bwMultiplier   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "QOS agent related fractional allocation
        of WAN bandwidth to WAN MC groups"
    ::= { bwMultiplierTableEntry 1 }

bwMultiplierTableIndex   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { bwMultiplierTableEntry 2 }

wanGroupsTable  OBJECT-TYPE
    SYNTAX  SEQUENCE OF WANGroupsTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { wanint 12 }

wanGroupsTableEntry     OBJECT-TYPE
    SYNTAX  WANGroupsTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX     { wanGroupsTableIndex }
    ::= { wanGroupsTable 1 }

WANGroupsTableEntry ::= SEQUENCE { wanGroupsTableIndex INTEGER,
                wanGroups1 INTEGER,
              wanGroups2 INTEGER }

wanGroups1   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanGroupsTableEntry 1 }

wanGroups2   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanGroupsTableEntry 2 }

wanGroupsTableIndex     OBJECT-TYPE
```

```
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanGroupsTableEntry 3 }

wanifCurStatus    OBJECT-TYPE
    SYNTAX  INTEGER (0..10)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "status number"
    ::= { wanint 13 }

wanifNumberSites OBJECT-TYPE
    SYNTAX  INTEGER (0..32)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanint 14 }

wanifNumberGroups  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanint 15 }

wanifInterfaceAddr    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanint 16 }

wanifGroupBaseAddr OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanint 17 }

wanifCtrlSocket   OBJECT-TYPE
    SYNTAX  INTEGER (0..10)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "status value"
    ::= { wanint 18 }
```

```
wanifDataSocket   OBJECT-TYPE
    SYNTAX  INTEGER (0..10)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "status value"
    ::= { wanint 19 }

wanifOutDataPacket   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "output data packets from the wan interface"
    ::= { wanint 20 }

wanifOutDataBytes    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "output data bytes from the wan interface"
    ::= { wanint 21 }

wanifOutCntlPacket   OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "number of control packets"
    ::= { wanint 22 }

wanifOutCntlBytes    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "number of bytes in the output contol
        packets.  Includes UDP/IP headers,
        not link layer"
    ::= { wanint 23 }

wanifMibTable    OBJECT-TYPE
    SYNTAX  SEQUENCE OF WanifMibTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { wanint 24 }

wanifMibTableEntry   OBJECT-TYPE
    SYNTAX  WanifMibTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { wanifGroupIndex }
```

```
    ::= { wanifMibTable 1 }

WanifMibTableEntry ::= SEQUENCE { wanifGroupIndex INTEGER,
                wanifOutputnPackets INTEGER }

wanifOutputnPackets  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanifMibTableEntry 1 }

wanifGroupIndex  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { wanifMibTableEntry 2 }

tempint1    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 25 }

tempint2    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 26 }

tempint3    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 27 }

tempint4    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
```

```
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 28 }

tempint5   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 29 }

tempint6   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 30 }

tempint7   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 31 }

tempint8   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 32 }

tempint9   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
```

```
        list on the fly"
    ::= { wanint 33 }

tempint10  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 34 }

tempint11  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 35 }

tempint12  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 36 }

tempint13  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 37 }

tempint14  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { wanint 38 }

tempint15  OBJECT-TYPE
```

```
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { wanint 39 }

    tempint16  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { wanint 40 }

    tempint17  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { wanint 41 }

    tempint18  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { wanint 42 }

    tempint19  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { wanint 43 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
```

```
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { wanint 44 }
END
--
************************************************************************
--
--   1.0    95/10/09 R. K. Nair
--          The QOS MIB - A part of HPAG Software
--
--   1.2    95/10/27 R. K. Nair
--       New variables added for ED1 A
--
************************************************************************
/


HPAG-QOS-MIB DEFINITIONS ::= BEGIN

    IMPORTS      Counter
                    FROM RFC1155-SMI
                 OBJECT-TYPE
                    FROM RFC-1212
                 DisplayString
                    FROM RFC1213-MIB
                 TRAP-TYPE
                    FROM RFC1215;

    nrl  OBJECT IDENTIFIER ::= { enterprises 394 }
    ritn OBJECT IDENTIFIER ::= { nrl 2 }
    hpag    OBJECT IDENTIFIER ::= { ritn 1 }
    qosOBJECT IDENTIFIER ::= { hpag 5 }

    -- The system group contains general information about the
    -- application software.

    sysDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-only
        STATUS  mandatory
            DESCRIPTION
                "A textual description of the entity.  This value
             should include the full name and version
                identification of the application software,
             software operating-system, and other
                descriptive text. It is mandatory that this only contain
                printable ASCII characters."
            ::= { qos 1 }

    sysContact OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-write
```

```
        STATUS  mandatory
        DESCRIPTION
            "The textual identification of the contact person
            for this application software, together with information
            on how to contact this person."
        ::= { qos 2 }

qosCalcMultiTab  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "if 1, recalculate qos table"
    ::= { qos 3 }

qosMultiTabValid  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "if 1, qos table valid"
    ::= { qos 4 }

qosRsvpReserveFail  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "if 0, RSVP reservation is successfull"
    ::= { qos 5 }

qosMibTable  OBJECT-TYPE
    SYNTAX  SEQUENCE OF QosMibTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { qos 6 }

qosMibTableEntry OBJECT-TYPE
    SYNTAX  QosMibTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { qosMultiplierIndex }
    ::= { qosMibTable 1 }

QosMibTableEntry ::= SEQUENCE { qosMultiplierIndex INTEGER,
                qosMultiplier INTEGER }

qosMultiplier  OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qosMibTableEntry 1 }
```

```
qosMultiplierIndex     OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qosMibTableEntry 2 }

qosResetTestFlowSpec    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qos 7 }

qosTestFlowSpecPktSize OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qos 8 }

qosTestFlowSpecPktRate OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qos 9 }

qosTestFlowSpecPktBurst    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { qos 10 }

tempint1    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 11 }

tempint2    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
```

```
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 12 }

    tempint3    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 13 }

    tempint4    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 14 }

    tempint5    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 15 }

    tempint6    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 16 }

    tempint7    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
```

```
        list on the fly"
    ::= { qos 17 }

tempint8    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 18 }

tempint9    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 19 }

tempint10   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 20 }

tempint11   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 21 }

tempint12   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { qos 22 }

tempint13   OBJECT-TYPE
```

```
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 23 }

    tempint14  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 24 }

    tempint15  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 25 }

    tempint16  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 26 }

    tempint17  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { qos 27 }

    tempint18  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
```

```
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { qos 28 }

    tempint19  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { qos 29 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { qos 30 }
END
```

APPENDIX D - APPLICATION TRANSLATOR (AT) MIB

```
/
**************************************************************************
--
--   1.0    95/05/05 R. K. Nair
--          Application Translator(AT) MIB
--
--   1.0    95/10/31 R. K. Nair
--       AT MIB modifications for ED 1 A
--
**************************************************************************
/


AT-MIB DEFINITIONS ::= BEGIN

   IMPORTS      Counter
                   FROM RFC1155-SMI
                OBJECT-TYPE
                   FROM RFC-1212
                DisplayString
                   FROM RFC1213-MIB
                TRAP-TYPE
                   FROM RFC1215;

   nrl OBJECT IDENTIFIER ::= { enterprises 394 }
   ritn OBJECT IDENTIFIER ::= { nrl 2 }
   at   OBJECT IDENTIFIER ::= { ritn 2 }
   ati  OBJECT IDENTIFIER ::= { at 2 }

   -- The system group contains general information about the
   -- application software.

   sysDescr OBJECT-TYPE
       SYNTAX  DisplayString (SIZE (0..255))
       ACCESS  read-only
       STATUS  mandatory
           DESCRIPTION
               "A textual description of the entity.  This value
            should include the full name and version
               identification of the application software,
            software operating-system, and other
               descriptive text. It is mandatory that this only contain
               printable ASCII characters."
           ::= { ati 1 }

   sysContact OBJECT-TYPE
           SYNTAX  DisplayString (SIZE (0..255))
```

```
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "The textual identification of the contact person
            for this application software, together with information
            on how to contact this person."
        ::= { ati 2 }

lgInPkts    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Legacy input packets per second"
    ::= { ati 3 }

lgOutPkts  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "Legacy output packets per second"
    ::= { ati 4 }

lgInBytes  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "byte counts include UDP/IP"
    ::= { ati 5 }

lgOutBytes OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "byte count"
    ::= { ati 6 }

ngInPkts    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "3.X input packets per second"
    ::= { ati 7 }

ngOutPkts OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "3.X output packets per second"
```

```
        ::= { ati 8 }

    ngInBytes OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "3.X bytes/second"
        ::= { ati 9 }

    ngOutBytes    OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "3.X bytes/second"
        ::= { ati 10 }

    pdusDropped2X   OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "Number of PDUs dropped - drops for any
            reason; wron ex. id., collisions, etc.
            Initial packet parse"
        ::= { ati 11 }

    collisionCount2X  OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "count of all local collision PDUs.  These
            PDUs are culled because they are of no outside
            interest"
        ::= { ati 12 }

    entityCount2X OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "total number of 2.X entities"
        ::= { ati 13 }

    qeCount2X       OBJECT-TYPE
        SYNTAX  INTEGER (0..2147483647)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
            "total number of entities on 2.X LAN"
        ::= { ati 14 }
```

```
thresholdLevel OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "defines sensitivity of QES"
    ::= { ati 15 }

qesStatus  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "qes ON/OFF status"
    ::= { ati 16 }

reduction  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "percent reduction with QES on"
    ::= { ati 17 }

bundling  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "bundling ON/OFF status"
    ::= { ati 18 }

bundleDelay  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "ms delay for bundling"
    ::= { ati 19 }

avePktsBundle OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "average packets per bundle"
    ::= { ati 20 }

pdusDropped3X  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Drops for any reason: wrong ex. id.,
```

```
        collisions, etc."
    ::= { ati 21 }

ngCInPkts OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Control Packets"
    ::= { ati 22 }

ngDInPkts OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Data Packets"
    ::= { ati 23 }

tempint4   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 24 }

tempint5   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 25 }

collisionCount3X  OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Count of all local collision PDUs.  These
        PDUs are culled because they are of no
        interest to the 2.X LAN"
    ::= { ati 26 }

remoteEntityCount OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
```

```
        "Total number of 3.X entities that the AT
        passes on to the 2.X LAN"
    ::= { ati 27 }

remoteQeCount    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Total number of 3.X entities which are
        quiescent. (This number is contained in
        remoteEntityCount"
    ::= { ati 28 }

tempint7   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 29 }

tempint8   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 30 }

tempint9   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 31 }

tempint10  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 32 }
```

```
tempint11  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 33 }

tempint12  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 34 }

tempint13  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 35 }

tempint14  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 36 }

tempint15  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ati 37 }

tempint16  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
```

```
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { ati 38 }

    tempint17  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { ati 39 }

    tempint18  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { ati 40 }

    tempint19  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { ati 41 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { ati 42 }

    overloadtrap   OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "ll overload condition"
        ::= { ati 43 }
```

```
    overloadcondition TRAP-TYPE
        ENTERPRISE ati
        VARIABLES  { overloadtrap }
        DESCRIPTION
            "ll overload condition"
        ::= 0

END
```

APPENDIX E - AGENT HOST (AH) MIB

```
--
*******************************************************************************
--
--   1.0    95/05/15 R. K. Nair
--          The Subscription Agent MIB - A part of AH Software
--
--   1.1    95/10/04 R. K. Nair
--      Subscription Agent MIB changes in preparation for ED1
--
*******************************************************************************
/


AH-SA-MIB DEFINITIONS ::= BEGIN

    IMPORTS      Counter
                    FROM RFC1155-SMI
                 OBJECT-TYPE
                    FROM RFC-1212
                 DisplayString
                    FROM RFC1213-MIB
                 TRAP-TYPE
                    FROM RFC1215;

    nrl  OBJECT IDENTIFIER ::= { enterprises 394 }
    ritn OBJECT IDENTIFIER ::= { nrl 2 }
    ah   OBJECT IDENTIFIER ::= { ritn 3 }
    sa   OBJECT IDENTIFIER ::= { ah 2 }

    -- The system group contains general information about the
    -- application software.

    sysDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-only
        STATUS  mandatory
            DESCRIPTION
                "A textual description of the entity.  This value
             should include the full name and version
                identification of the application software,
             software operating-system, and other
                descriptive text. It is mandatory that this only contain
                printable ASCII characters."
            ::= { sa 1 }

    sysContact OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                "The textual identification of the contact person
```

```
            for this application software, together with information
            on how to contact this person."
        ::= { sa 2 }

principals  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "current number of client subagents (customers)
        Each ModSAF backend counts as 1"
    ::= { sa 3 }

principalCreates    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "cumulative counter of number of principals
        created"
    ::= { sa 4 }

principalDeletes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "cumulative counter of number of principals
        deleted"
    ::= { sa 5 }

numLocalEntities  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "current number of locally generated entities.
        Sum of numLocalEntities over all 7 sites
        gives total num entities in exercise"
    ::= { sa 6 }

entityCreates    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Cumulative counter of all of the entities
        the AH sees. This includes all of the entities
        generated on this LAN plus all of the entities
        from remote sites which show up on this LAN"
    ::= { sa 7 }

entityTimeouts OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
```

```
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Cumulative counter of number of entities which
        the AH times out - locally and remotely generated
        entities"
    ::= { sa 8 }

quiescentEntities  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "current number of quiescent entities (locally
        and remotely generated)"
    ::= { sa 9}

quiescentDeclares  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "cumulative counter of number of quiescent
        declarations (locally and remotely generated)"
    ::= { sa 10 }

activeDeclares OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Cumulative counter of number of active
        declarations (locally and remotely generated)"
    ::= { sa 11 }

tempint1   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 12 }

tempint2   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 13 }
```

```
tempint3    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 14 }

tempint4    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 15 }

tempint5    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 16 }

tempint6    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 17 }

tempint7    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 18 }

tempint8    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
```

```
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 19 }

    tempint9    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 20 }

    tempint10  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 21 }

    tempint11  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 22 }

    tempint12  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 23 }

    tempint13  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
```

```
        list on the fly"
    ::= { sa 24 }

tempint14  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 25 }

tempint15  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 26 }

tempint16  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 27 }

tempint17  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 28 }

tempint18  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { sa 29 }

tempint19  OBJECT-TYPE
```

```
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 30 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
             user could add new variables to the managed
             list on the fly"
        ::= { sa 31 }
END
--
****************************************************************************
--
--   1.0    95/05/12 R. K. Nair
--          The Consistency Agent MIB - A part of AH Software
--
--   1.1    95/10/04 R. K. Nair
--       Consistency Agent MIB changes in preparation for ED1
--
****************************************************************************
/


AH-CA-MIB DEFINITIONS ::= BEGIN

    IMPORTS      Counter
                    FROM RFC1155-SMI
                 OBJECT-TYPE
                    FROM RFC-1212
                 DisplayString
                    FROM RFC1213-MIB
                 TRAP-TYPE
                    FROM RFC1215;

    nrl OBJECT IDENTIFIER ::= { enterprises 394 }
    ritn OBJECT IDENTIFIER ::= { nrl 2 }
    ah  OBJECT IDENTIFIER ::= { ritn 3 }
    ca  OBJECT IDENTIFIER ::= { ah 3 }

    -- The system group contains general information about the
    -- application software.

    sysDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
```

```
        ACCESS  read-only
        STATUS  mandatory
            DESCRIPTION
                "A textual description of the entity.  This value
            should include the full name and version
                identification of the application software,
            software operating-system, and other
                descriptive text. It is mandatory that this only contain
                printable ASCII characters."
            ::= { ca 1 }

    sysContact OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                "The textual identification of the contact person
                for this application software, together with information
                on how to contact this person."
            ::= { ca 2 }

    inputPackets    OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { ca 3 }

    inputBytes OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "byte counts include UDP/IP headers
            but not link level headers"
        ::= { ca 4 }

    inEntityStatePDUs OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { ca 5 }

    inCreateDataPDUs OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            ""
        ::= { ca 6 }
```

```
inDeleteDataPDUs OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 7 }

inUpdateDataPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 8 }

inFullDataPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 9 }

inRequestDataPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 10 }

inRequestGroupPDUs OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 11 }

inTargetedRefreshPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 12 }

inGroupRefreshPDUs OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
```

```
    ::= { ca 13 }

inDPAdvertisementPDUs OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 14 }

inEntityStateBytes OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 15 }

inCreateDataBytes OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 16 }

inDeleteDataBytes OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 17 }

inUpdateDataBytes    OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 18 }

inFullDataBytes    OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
    DESCRIPTION
        ""
    ::= { ca 19 }

inRequestDataBytes    OBJECT-TYPE
    SYNTAX INTEGER (-2147483647..2147483647)
    ACCESS read-write
    STATUS mandatory
```

```
    DESCRIPTION
        ""
    ::= { ca 20 }

inRequestGroupBytes OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 21 }

inTargetedRefreshBytes   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 22 }

inGroupRefreshBytes OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 23 }

inDPAdvertisementBytes  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 24 }

outputPackets  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 25 }

outputBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "byte counts include UDP/IP headers,
        but not link level headers"
    ::= { ca 26 }

outCreateDataPDUs    OBJECT-TYPE
```

```
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 27 }

outDeleteDataPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 28 }

outUpdateDataPDUs  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 29 }

outFullDataPDUs  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 30 }

outRequestDataPDUs  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 31 }

outRequestGroupPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 32 }

outTargetedRefreshPDUs OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 33 }
```

```
outGroupRefreshPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 34 }

outDPAdvertisementPDUs    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 35 }

outCreateDataBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 36 }

outDeleteDataBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 37 }

outUpdateDataBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 38 }

outFullDataBytes  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 39 }

outRequestDataBytes  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
```

```
        ""
    ::= { ca 40 }

outRequestGroupBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 41 }

outTargetedRefreshBytes  OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 42 }

outGroupRefreshBytes    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 43 }

outDPAdvertisementBytes OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 44 }

uptime OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 45 }

multicastGroupsInuse OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 46 }

multicastJoins OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
```

```
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 47 }

multicastLeaves    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        ""
    ::= { ca 48 }

inputPDUTable  OBJECT-TYPE
    SYNTAX  SEQUENCE OF InputPDUTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { ca 49 }

inputPDUTableEntry    OBJECT-TYPE
    SYNTAX  InputPDUTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { inputPDUType }
    ::= { inputPDUTable 1 }

InputPDUTableEntry ::= SEQUENCE { inputPDUType INTEGER,
                inputPacketCount INTEGER,
                inputByteCount INTEGER }

inputPacketCount    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "input PDU Packet Counts by type "
    ::= { inputPDUTableEntry 1 }

inputByteCount    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "input PDU Byte Counts by type "
    ::= { inputPDUTableEntry 2 }

inputPDUType OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "input PDU type"
    ::= { inputPDUTableEntry 3 }
```

```
outputPDUTable  OBJECT-TYPE
    SYNTAX  SEQUENCE OF OutputPDUTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { ca 50 }

outputPDUTableEntry     OBJECT-TYPE
    SYNTAX  OutputPDUTableEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { outputPDUType }
    ::= { outputPDUTable 1 }

OutputPDUTableEntry ::= SEQUENCE { outputPDUType INTEGER,
                outputPacketCount INTEGER,
                outputByteCount INTEGER }

outputPacketCount    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "output PDU Packet Counts by type "
    ::= { outputPDUTableEntry 1 }

outputByteCount    OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "output PDU Byte Counts by type "
    ::= { outputPDUTableEntry 2 }

outputPDUType OBJECT-TYPE
    SYNTAX  INTEGER (0..2147483647)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "output PDU type"
    ::= { outputPDUTableEntry 3 }

tempint1    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 51 }

tempint2    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
```

```
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 52 }

tempint3   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 53 }

tempint4   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 54 }

tempint5   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 55 }

tempint6   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 56 }

tempint7   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
```

```
        list on the fly"
    ::= { ca 57 }

tempint8    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 58 }

tempint9    OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 59 }

tempint10   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 60 }

tempint11   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 61 }

tempint12   OBJECT-TYPE
    SYNTAX  INTEGER (-2147483647..2147483647)
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A temporary variable - specified so that
        user could add new variables to the managed
        list on the fly"
    ::= { ca 62 }

tempint13   OBJECT-TYPE
```

```
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 63 }

    tempint14  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 64 }

    tempint15  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 65 }

    tempint16  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 66 }

    tempint17  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 67 }

    tempint18  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
```

```
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 68 }

    tempint19  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 69 }

    tempint20  OBJECT-TYPE
        SYNTAX  INTEGER (-2147483647..2147483647)
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
            "A temporary variable - specified so that
            user could add new variables to the managed
            list on the fly"
        ::= { ca 70 }
END
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1996	Final : 14–16 November 1995

4. TITLE AND SUBTITLE

SYNTHETIC THEATER OF WAR (STOW) ENGINEERING DEMONSTRATION-1A (ED-1A) ANALYSIS REPORT

5. FUNDING NUMBERS

PE: 0603226E
AN: DN301038
WU: 441-CC28

6. AUTHOR(S)

T. R. Tiernan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division
San Diego, California 92152–5000

8. PERFORMING ORGANIZATION REPORT NUMBER

TR 1722

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The primary goal of the Real-Time Information Transfer and Networking (RITN) effort is to evaluate and develop technologies to integrate into a scaleable network solution for the distribution simulation ACTD STOW 97. Prior to ED-1A, the greatest success for network solutions within the STOW program occurred with the demonstration of the Synthetic Theater of War-Europe (STOW-E) in November 1994, in which 1800 simulation entries were supported on a distributed network with a 1.1 Mbps throughput limitation. The entity count goal to demonstrate the success of RITN in Engineering Demonstration 1A (ED-1A) was 5000 entities. Through the course of the demonstration, RITN achieved all of its major objectives. This report discusses several conclusions. Multicasting reduced the volume of traffic received by individual sites by 35% to 60% over that which would have been received with a broadcast delivery scheme. The new bi-level multicast and consistency protocols were very robust and show promise for extensibility into future applications. The High Performance Application Gateways (HPAG) introduced minimal transmission delays while providing a router interface between the LAN and WAN not available in the commercial world. The agent architecture employed was effective and contributed minimal additional traffic to the network. The RITN architecture allows for the participation of legacy simulations in the large-scale exercises of the future. An exercise of the magnitude envisioned for STOW 97 can be supported by existing backbone technology. LAN technologies must be upgraded to support local data loads to host processors. The Simple Network Management Protocol (SNMP), with minor adjustments to the ED-1A implementation, will be an effective and indispensable tool for exercise management and rapid data collection in future exercises.

14. SUBJECT TERMS

Mission Area: Command, Control, Communications
networking
simulation
distributed processing
warfighter training

15. NUMBER OF PAGES

209

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

21a. NAME OF RESPONSIBLE INDIVIDUAL	21b. TELEPHONE *(include Area Code)*	21c. OFFICE SYMBOL
T. R. Tiernan	(619) 553–3562	Code 33562

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 0271	Archive/Stock	(6)
Code 0274	Library	(2)
Code 44	J. D. Grossman	(1)
Code 441	T. R. Tiernan	(1)
Code 44203	C. B. Peters	(24)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (4)

NCCOSC Washington Liaison Office
Arlington, VA 22245–5200

Center for Naval Analyses
Alexandria, VA 22302–0268

Navy Acquisition, Research and Development
Information Center (NARDIC)
Arlington, VA 22244–5114

GIDEP Operations Center
Corona, CA 91718–8000